X3T11/97-
X3T11/Project 1213-D/REV 1.2

HIGH-PERFORMANCE PARALLEL INTERFACE - 6400 Mbit/s Physical Layer

(HIPPI-6400-PH)

April 1, 1997

Secretariat:

Information Technology Industry Council (ITI)

ABSTRACT: This standard specifies a physical-level, point-to-point, full-duplex, link interface for reliable, flow-controlled, transmission of user data at 6400 Mbit/s, per direction, across distances of 1 km. A parallel copper cable interface for distances of up to 50 m is specified. Connections to a separate longer-distance optical interface are provided. Small fixed-size micropackets provide an efficient, low-latency, structure for small transfers, and a component for large transfers.

NOTE:

This is an internal working document of X3T11, a Technical Committee of Accredited Standards Committee X3. As such, this is not a completed standard. The contents are actively being modified by X3T11. This document is made available for review and comment only. For current information on the status of this document contact the individuals shown below:

POINTS OF CONTACT:

Roger Cummings (X3T11 Chairman)
Distributed Processing Technology
140 Candace Drive
Maitland, FL 32751
(407) 830-5522 x348, Fax: (407) 260-5366
E-mail: cummings_roger@dpt.com

Ed Grivna (X3T11 Vice-Chairman)
Cypress Semiconductor
2401 East 86th Street
Bloomington, MN 55425
(612) 851-5200, Fax: (612) 851-5087
E-mail: elg@cypress.com

Don Tolmie (HIPPI-6400-PH Technical Editor)
Los Alamos National Laboratory
CIC-5, MS-B255
Los Alamos, NM 87545
(505) 667-5502, Fax: (505) 665-7793
E-mail: det@lanl.gov

Comments on Rev 1.2

This is a preliminary document undergoing lots of changes. Many of the additions are just place holders, or are put there to stimulate discussion. Hence, do not assume that the items herein are correct, or final – everything is subject to change. This page tries to outline where we are; what has been discussed and semi-approved, and what has been added or changed recently and deserves your special attention. This summary relates to changes since the previous revision. Also, previous open issues are outlined with a single box, new open issues ones are marked with a double bar on the left edge of the box.

Changes are marked with margin bars so that changed paragraphs are easily found, and then highlights mark the specific changes. The list below just describes the major changes, for detail changes please compare this revision to the previous revision.

Please help us in this development process by sending comments, corrections, and suggestions to the Technical Editor, Don Tolmie, of the Los Alamos National Laboratory, at det@lanl.gov. If you would like to address the whole group working on this document, send the Message to hippi@network.com.

1. Changed the first paragraph of the Introduction to remove the specification of a fiber interface in the document, and make it match the Abstract and Foreword.
2. In the Introduction, changed the next to last bullet from "…for limited distance applications over parallel copper cable." to "for driving parallel copper cable over limited distances.". Changed the last bullet from "A parallel low-level electrical interface…" to "An ac-coupled parallel electrical interface…". Deleted the bullets stating "Support for carrying legacy HIPPI-800 and HIPPI-1600 traffic." and "A physical-layer interface continuing the HIPPI tradition of simplicity.".
3. In clause 2, added normative references for HIPPI-ST and HIPPI-6400-OPT.
4. In 3.1.14, added a definition for "station management". Also added "SMT" to the list of acronyms in 3.3.
5. In 3.1.15, changed the definition of ULA to "A logical address stored in a Source or Destination field that uniquely identifies and Originating Source or Final Destination. The ULA conforms to…". This makes the definition consistent with the one in HIPPI-6400-SC.
6. In 3.3, changed the abbreviation of ULA by removing "MAC".
7. In 4, deleted the last of the last sentence, which read "…system, with all of the HIPPI-6400 connections using copper cable.".
8. Swapped the original order of 4.9 and 4.10 to agree with the order of the referenced clauses 15 and 16. Did major rewording to 4.9, e.g., "…local electrical interface…" to "…local on-board electrical interface…", just referenced HIPPI-6400-OPT instead of saying what it contained, and stated that the distance limit was a just a function of TSEQ and RSEQ and didn't mention optical limitations.
9. Added all of the service interface in clauses 5.3 through 5.5. The text was only margin barred – it wasn't highlighted since that can make it harder to read. All of this text needs to be reviewed carefully, this is the first time it has seen the light of day and will probably undergo quite a few changes before it is finalized.
10. In 6.2, expanded the definition of VC3 to state the maximum number of micropackets it could contain.
11. A major change, but one using few words, was in 7. Here the words in the last sentence of the first paragraph were changed from "…shall be padded in the last micropacket." to "…shall be padded with zeros in the last micropacket.".
12. In 7.1, changed "…(Header micropacket)…" to "…(TYPE = Header)…".
13. In 8.4, changed the note from "This value…" to "The number of allowable retransmissions…" and changed "…noise hits" to "noise hits or short cables".
14. In figure 18, changed the shape of the CLOCK signal during the training sequence.

15. In 11.2, changed the "training timer" to "Dead-man timer", and changed "Skew_Training_Reset_Error" to "Reset_Intialize_Error". In the second paragraph, changed "If the skew-compensation circuitry fails…" to "If the periodic retraining sequences fail…".
16. In 12, changed "…may trigger…" to "…shall trigger…". Changed "…system dependent" to "…system dependent, and is independent of the Hold-off timer (see 12.3)."
17. In 12.1, first paragraph, changed "…and the Hold-off timer is expired or not running (see 12.3)" to "…Power-on = true, and a Link Reset sequence is not currently in progress…"; changed "…may…" to "…shall…". In the bullets, changed it so it is "may" for the system administrator, and "shall" for the others. Changed "Port_Reset Admin…" to "RESET Admin…"; changed "…Training timer…" to "…Dead-man timer…". Moved the item about Link Reset being triggered by power-on from a bullet to a separate paragraph. Changed the bullet "…shall not be modified except when the sequence was triggered by a system power-on transition" to "…shall be initialized if the Link Reset was triggered by a system power-on transition, otherwise they shall not be modified. Changed "At exit, the local state shall be reset with:" to "At exit:". Changed "The Disparity Count will have been corrected with the training sequences" to "and the Disparity Count shall be accurate".
18. In 12.2, changed the first sentence to "…Power-on = true, and Initialize sequence is not currently in progress (see figure 19), and the Hold-off timer is expired or not running (see 12.3), then …". Changed it so that it is "may" for the system administrator, and "shall" for the other case. In the third paragraph, changed "…Training timer…" to "…Dead-man timer period (see 11.2 and table 6).". In the last paragraph, changed the bullets to in-line text reading "and all of the timeout timers (see table 6), except for the Hold-off timer, shall be initialized.".
19. In figure 19, changed "Training timer" to "Dead-man timer" in 5 places. Merged the "Reset local state" and "Start Dead-man timer" into one state. Deleted the states checking for Initialize and Reset micropackets being received just before dropping into the Normal operation state. Fixed up some of the interconnection lines.
20. In 12.3, changed "…between…" to "…among…".
21. In 13.1, the first paragraph was shortened by deleting the text describing how the optical system, or copper system, would implement the activity monitor. The activity monitor = true bullets were moved ahead of the = false case. The descriptions of what happens when = false was changed to "… the transmitting and receiving circuits may be disabled to prevent damage to the interface or personnel (see 15.3 and 16.3)". The statement about the signals driving the activity monitor was deleted.
22. In 13.2, the first line was changed from "…may…" to "…shall…". In the bullet starting "The receiver shall discard…" micropackets with TYPEs = Reset and Reset_ACK were added. In the last paragraph, "…local management…" was changed to "…local administrator…".
23. The order was changed on 14.1 and 14.2 to match where the tables could be placed. In 14.2, the word "error" was changed to "event" in three places.
24. In table 6, "Training timer" was changed to "Dead-man timer".
25. In table 7, "Skew_Training_Reset_Error" was changed to "Reset_Initialize_Error".
26. In clause 15, there were lots of changes, mainly to make the local and copper interfaces consistent. Table 8 was added to specify the component values, and figure 20 was added to define the interface. All of 15.2 and 15.3 is new.
27. Table 9 was a copy of the original timing table, but with the 16-bit column removed. The percentage values were corrected.
28. Tables 10 was modified with the note about single-ended measurements, jitter was changed to "peak-to-peak" jitter, and "channel-to-channel" was replaced with "pair-to-pair".
29. Table 11 was changed to be consistent with table 15, and some of the names updated.
30. Clause 16 was updated to be consistent with clause 15. The component values were called out in a table rather than in-line text. Figure 21 was changed by moving the signal

name to the driver output pins rather than after the equalizer network, the names were changed on the components to give a left-to-right order.

31. Table 13 was a copy of the original timing table, but with the 8-bit column removed. The percentage values were corrected.
32. Table 14 had some of the parameters corrected, and the note about single-ended measurements added.
33. In 16.4, changed "…bulkhead receptacle…" to "…receptacle…" two places in the clause. Changed "…90^o exit…" to "…straight exit…".
34. In table 16, changed "Imbalance skew" to "Imbalance skew within a signal pair". Added the note about the measurements being single-ended.
35. In figure 23, changed "Bulkhead receptacle" to "Receptacle".
36. In figure 24, changed the title from "Bulkhead receptacle" to "Receptacle". Changed the A16 dimension to "Dependent on board thickness".
37. In figure 25, swapped the dimensions for B6 and B8. Made the jackscrews more uniform in size.
38. In A.6, changed "coax" to "twin-ax", and added the reference to 16.1.
39. In table A.7, added the component values.

Contents

Tables

Figures

Annex

Foreword (This foreword is not part of American National Standard X3.xxx-199x.)

This American National Standard specifies a physical-level, point-to-point, full-duplex, link interface for reliable, flow-controlled, transmission of user data at 6400 Mbit/s, per direction, across distances of 1 km. A parallel copper cable interface for distances of up to 50 m is specified. Connections to a separate longer-distance optical interface are provided. Small fixed-size micropackets provide an efficient, low-latency, structure for small transfers, and a component for large transfers.

This standard provides an upward growth path for legacy HIPPI-based systems.

This document includes one annex which is informative and is not considered part of the standard.

Requests for interpretation, suggestions for improvement or addenda, or defect reports are welcome. They should be sent to the X3 Secretariat, Information Technology Industry Council, 1250 Eye Street, NW, Suite 200, Washington, DC 20005.

This standard was processed and approved for submittal to ANSI by Accredited Standards Committee on Information Processing Systems, X3. Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, the X3 Committee had the following members:

> (List of X3 Committee members to be included in the published standard by the ANSI Editor.)

Subcommittee X3T11 on Device Level Interfaces, which developed this standard, had the following participants:

> (List of X3T11 Committee members, and other active participants, at the time the document is forwarded for public review, will be included by the Technical Editor.)

Introduction

This American National Standard specifies a physical-level, point-to-point, full-duplex, link interface for reliable, flow-controlled, transmission of user data at 6400 Mbit/s, per direction, across distances of 1 km. A parallel copper cable interface for distances of up to 50 m is specified. Connections to a separate longer-distance optical interface are provided. Small fixed-size micropackets provide an efficient, low-latency, structure for small transfers, and a component for large transfers.

Characteristics of a HIPPI-6400-PH physical-layer interface include:

– User data transfer bandwidth of 6400 Mbit/s (800 MByte/s).

– A full-duplex link capable of independent full-bandwidth transfers in both directions simultaneously.

– Four virtual circuits providing a limited multiplexing capability.

– A fixed size transfer unit, i.e., a 32-byte micropacket, for hardware efficiency.

– A small transfer unit resulting in low latency for short Messages, and a component for large transfers.

– Credit-based flow control that prevents buffer overflow.

– End-to-end, as well as link-to-link, checksums.

– Automatic retransmission to correct flawed data providing guaranteed, in-order, reliable, data delivery.

– An ac coupled parallel electrical interface for driving parallel copper cable over limited distances.

– An ac coupled parallel electrical interface for driving a local optical interface for longer distances.

American National Standard for Information Technology –

High-Performance Parallel Interface – 6400 Mbit/s Physical Layer (HIPPI-6400-PH)

1 Scope

This American National Standard specifies a physical-level, point-to-point, full-duplex, link interface for reliable, flow-controlled, transmission of user data at 6400 Mbit/s, per direction, across distances of 1 km. A parallel copper cable interface for distances of up to 50 m is specified. Connections to a separate longer-distance optical interface are provided. Small fixed-size micropackets provide an efficient, low-latency, structure for small transfers, and a component for large transfers.

Specifications are included for:

– automatic retransmission to correct flawed data;

– the format of a small data transfer unit called a micropacket;

– a Message structure that includes routing information for network applications;

– end-to-end, as well as link-to-link, checksums;

– the timing requirements of the parallel signals;

– a parallel interface using copper coaxial cable;

– connections to a separate local optical interface;

– a link-level protocol tuned for a maximum distance of 1 km.

2 Normative references

The following American National Standards contain provisions which, through reference in this text, constitute provisions of this American National Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

ANSI X3.210-1992, *High-Performance Parallel Interface, Framing Protocol (HIPPI-FP)*

ANSI X3.xxx-199x, *High-Performance Parallel Interface, Scheduled Transfer (HIPPI-ST)*

ANSI X3.xxx-199x, *High-Performance Parallel Interface, 6400 Mbit/s Switch Control (HIPPI-6400-SC)*

ANSI X3.xxx-199x, *High-Performance Parallel Interface, 6400 Mbit/s Optical Interface (HIPPI-6400-OPT)*

ANSI/IEEE Std 802-1990, *IEEE Standards for Local and Metropolitan Area Networks: Overview and architecture (formerly known as IEEE Std 802.1A, Project 802: Local and Metropolitan Area Network Standard — Overview and Architecture)*

ISO/IEC 8802-2:1989 (ANSI/IEEE Std 802.2-1989), *Information Processing Systems – Local Area Networks – Part 2: Logical link control*

3 Definitions and conventions

3.1 Definitions

For the purposes of this standard, the following definitions apply.

3.1.1 acknowledge (ACK): Confirmation that the Destination has received the micropacket without errors.

3.1.2 administrator: A station management entity providing external management control.

3.1.3 credit: A credit corresponds to one micropacket's worth of buffer space available in the Destination's VC buffer.

3.1.4 Destination: The receiving end of a physical link.

3.1.5 Final Destination: The end device that receives, and operates on, the data payload portion of the micropackets. This is typically a host computer system, but may also be a non-transparent translator, bridge, or router.

3.1.6 link: A full-duplex connection between HIPPI-6400-PH devices.

3.1.7 log: The act of making a record of an event for later use.

3.1.8 Message: An ordered sequence of one or more micropackets that have the same VC, Originating Source, and Final Destination. Messages are the basic transfer unit between an Originating Source and a Final Destination. The first micropacket of a Message is a Header micropacket. The last micropacket, which may also be the first micropacket, has the TAIL bit set. (See 4.4.)

3.1.9 micropacket: The basic transfer unit, between a Source and Destination, consisting of 32 data bytes and 64 bits of control information.

3.1.10 optional: Characteristics that are not required by HIPPI-6400-PH. However, if any optional characteristic is implemented, it shall be implemented as defined in HIPPI-6400-PH.

3.1.11 Originating Source: The end device that generates the data payload portion of the micropackets. This is typically a host computer system, but may also be a non-transparent translator, bridge, or router.

3.1.12 Source: The sending end of a physical link.

3.1.13 station management (SMT): The supervisory entity that monitors and controls the HIPPI-6400-PH entity.

3.1.14 syndrome: The value (should be zero) obtained by exclusive ORing the calculated CRC value with the CRC value received with the micropacket.

3.1.15 Universal LAN MAC Address (ULA): A logical address stored in a Source or Destination field that uniquely identifies an Originating Source or Final Destination. The ULA conforms to the 48-bit MAC address specified by the IEEE 802 Overview Standard.

3.1.16 upper-layer protocol (ULP): The protocols above the service interface. These could be implemented in hardware, software, or they could be distributed between the two.

3.1.17 Virtual Channel (VC): One of four logical paths within each direction of a single link.

3.2 Editorial conventions

In this standard, certain terms that are proper names of signals or similar terms are printed in uppercase to avoid possible confusion with other uses of the same words (e.g., FRAME). Any lowercase uses of these words have the normal technical English meaning.

A number of conditions, sequence parameters, events, states, or similar terms are printed with the first letter of each word in uppercase and the rest lowercase (e.g., Block, Source). Any lowercase uses of these words have the normal technical English meaning.

The word *shall* when used in this American National standard, states a mandatory rule or requirement. The word *should* when used in this standard, states a recommendation.

3.2.1 Binary notation

Binary notation is used to represent relatively short fields. For example a two-bit field containing the binary value of 10 is shown in binary format as b'10'.

3.2.2 Hexadecimal notation

Hexadecimal notation is used to represent some fields. For example a two-byte field containing a binary value of b'11000100 00000011' is shown in hexadecimal format as x'C403'.

3.3 Acronyms and other abbreviations

ACK	acknowledge indication
CR	credit amount parameter
CRC	cyclic redundancy check
DSAP	Destination Service Access Protocol
ECRC	end-to-end CRC
HIPPI	High-Performance Parallel Interface
Hz	Hertz = 1 cycle per second
K	kilo (2^{10} or 1024)
LCRC	link CRC
LLC	Logical Link Control
lsb	least significant bit
M	mega (2^{20} or 1,048,576)
MAC	Media Access Control
ms	milliseconds
msb	most significant bit
ns	nanoseconds
p-p	peak to peak
ps	picoseconds
RSEQ	receive sequence number
SMT	station management
SNAP	SubNetwork Access Protocol
SSAP	Source Service Access Protocol
TSEQ	transmit sequence number
ULA	Universal LAN Address
ULP	upper-layer protocol
VC	Virtual Channel
VCR	Virtual Channel Credit selector
µs	microseconds
Ω	ohms

4 System overview

This clause provides an overview of the structure, concepts, and mechanisms used in HIPPI-6400-PH. Figure 1 gives an example of a HIPPI-6400 system.

4.1 Links

HIPPI-6400-PH defines a point-to-point physical link for transferring micropackets. The physical links, as shown in figure 2 between a local end and a remote end, are bi-directional. The logical links are simplex, i.e., the data inbound and outbound are completely separate. Some control information, e.g., credit, flows in the reverse direction, and it is included in the micropackets flowing in the reverse direction. This is why the physical links must be bi-directional with information flowing in both directions simultaneously.

A link is composed of two Sources that transmit information, and two Destinations that receive information. Each end of a link has a Source and a Destination.

The data path is 16 bits wide for the copper implementation, and is eight bits wide for a fiber implementation. The control path is one-fourth the width of the data path, e.g., the control path for the copper implementation is 4 bits wide. CLOCK, CLOCK_2, and FRAME are individual signals carried on separate conductors. The CLOCK_2 signal is only used in 16-bit systems.

4.2 Virtual Channels

Four Virtual Channels, VC0, VC1, VC2, and VC3 are available in each direction on each link. The VCs are assigned to specific Message sizes and transfer methods.

All of the micropackets of a Message are transmitted on a single VC, i.e., the VC number does not change as the micropackets travel from the Originating Source to the Final Destination over one or more links. Messages to a Final Destination are delivered in order on a single VC. Multiple messages may be out of order if sent over different VCs—even if the VCs are in the same physical link. The VCs provide a multiplexing mechanism which can be used to prevent a large Message from Blocking a small Message until the large Message has completed.

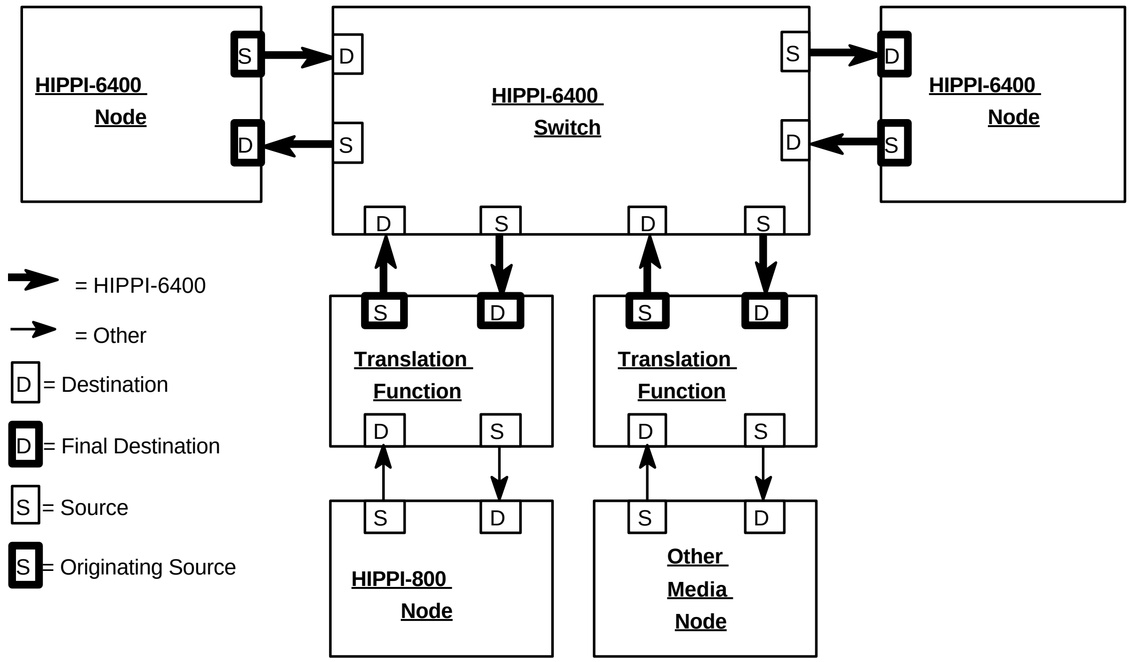


Figure 1 – System overview

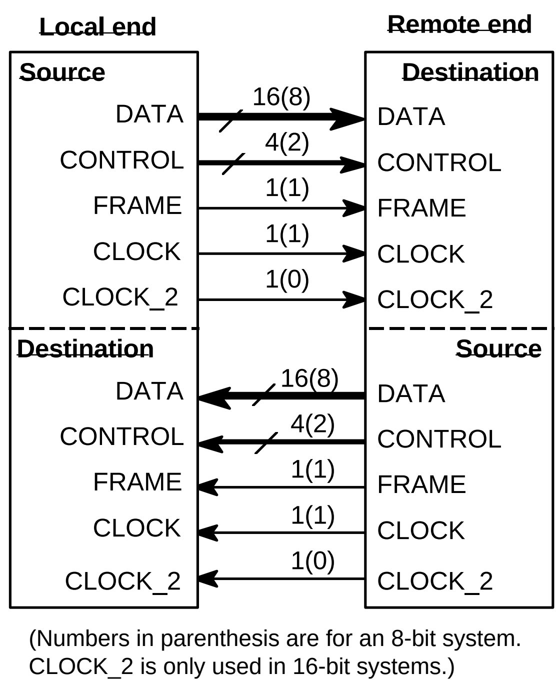


Figure 2 – HIPPI-6400-PH link showing signal lines

4.3 Micropacket

Micropackets are the basic transfer unit from Source to Destination on a link. As shown in figure 3 a micropacket is composed of 32 data bytes and 64 bits of control information. At 6400 Mbit/s a micropacket is transmitted every 40 ns, with Null micropackets transmitted when other micropackets are not available. Credit and retransmit operations are performed on a micropacket basis.

The 64 bits of control information in each micropacket includes parameters for:

– selecting a VC;

– detecting missing micropackets;

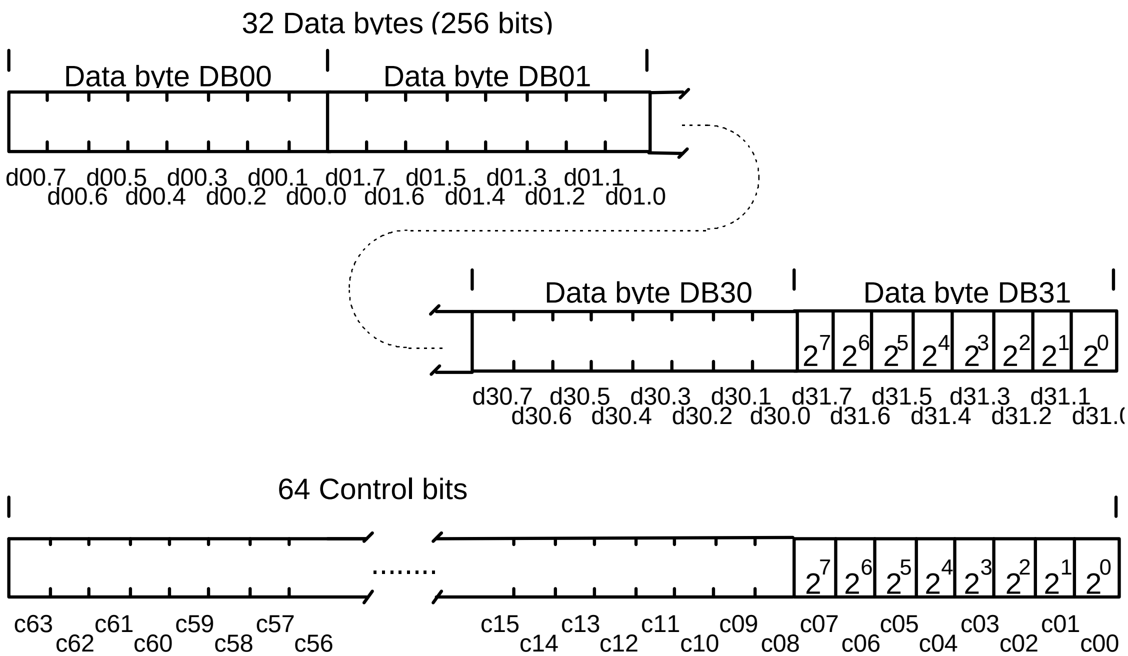
– denoting the types of information in the micropacket;

– marking the last micropacket of a Message;

– signaling that the Message was truncated at its originator, or damaged en-route;

– passing credit information from the Destination to the Source;

– Link-level and end-to-end checksums.



Naming conventions:

Data bytes are labeled capital DB and a two-digit number, e.g., DB00.
In a parameter that uses multiple bytes, the most-significant byte is the lowest-numbered byte.
Data bits are labeled lower case d, a two-digit byte number, and a one-digit bit number, e.g., d31.7.
Control bits are labeled lower case c and a two-digit number, e.g., c00.
In a parameter that uses multiple bits, the most-significant bit is the highest-numbered bit.

Figure 3 – Logical micropacket format and naming conventions

4.4 Message

As shown in figure 4, a Message is an ordered sequence of one or more micropackets which have the same VC, Originating Source, and Final Destination. The first micropacket of a Message, i.e., the Header micropacket, contains information used to route through a HIPPI-6400 fabric. The last micropacket of the Message is marked with the TAIL bit.

Micropacket Transmission order		
1	Header information	c63–c00
2	1st 32 bytes of Mesage data	c63–c00
3	2nd 32 bytes of Message data	c63–c00
⋮	⋮	
n	Last bytes of Message data	c63–c00

Figure 4 – Message format

4.5 FRAME and CLOCK signals

The FRAME signal, carried on a separate signal line, marks a micropacket's beginning. Both edges of either the CLOCK or CLOCK_2 signals, also carried on separate signal lines, are used for strobing the data. The data, control, and FRAME signals from a Source are synchronous with that Source's CLOCK and CLOCK_2 signals. The CLOCK rate is dependent on the width of the data bus, e.g., a 16-bit data bus utilizing 4b/5b encoding requires the CLOCK line to run at 250 MHz and each data and control line may transition every 2 ns.

4.6 Flow control

Link-level credit-based flow control is used between a Source and Destination. As shown in figure 5, the credits are assigned on a VC basis, i.e., VC0's credits are separate from VC1's credits. The Destination end of a link grants credits to match the number of free receive buffers for a particular VC. The Source end of the link consumes credits as it moves micropackets from the VC Buffers to the Output Buffer. Note that flow control is on a link basis, i.e., hop-by-hop.

4.7 Retransmission

Retransmission is performed to correct flawed micropackets (see 8.4). Go-back-N retransmission is used, i.e., if an error is detected then the flawed micropacket, and all micropackets transmitted after it, are retransmitted. The CRCs in each micropacket are checked at the Destination side of a link; at the Input Buffer in figure 5. Correct micropackets are acknowledged, flawed micropackets are discarded. Note that retransmission is independent of the VC used, and also independent of the credit information, i.e., retransmission occurs between the Output and Input Buffers in figure 5 while VC and credit information pertains only to the VC Buffers. Retransmission is on a link basis, i.e., hop-by-hop.

4.8 Check functions

As shown in table 1, two 16-bit cyclic redundancy checks (CRCs) are used, and they use different polynomials. The end-to-end CRC (ECRC) covers the data bytes of all of the micropackets in a Message, i.e., the Header micropacket and all of the Data micropackets (if any) up to this point in a Message. The ECRC does not cover the control bits. The ECRC is unchanged from the Originating Source to the Final Destination. The ECRC is accumulated over an entire Message, i.e., it is not re-initialized for intermediate Data micropackets (see 6.6.3). Note that in table 1, the second micropacket's ECRC covers the information in the first and second micropacket; the third micropacket's ECRC covers the information in the first, second, and third micropacket, etc.

The link CRC (LCRC) covers all of the data and control bits of a micropacket, with the exception of itself. The LCRC is initialized for each micropacket, and must be calculated fresh for each link since other control fields change.

The combination of two 16-bit CRCs provides a stronger check than a single 16-bit CRC for link-level checking of individual micropackets. In addition, the 16-bit ECRC provides checking over a whole Message.

4.9 Local electrical interface (optional)

The optional local on-board electrical interface (see clause 15) provides a connection to a separately specified optical interface (see HIPPI-6400-OPT) for longer distances. Note that the TSEQ and RSEQ parameter sizes in this standard allow full speed operation at distances up to 1 km.

4.10 Copper cable physical layer (optional)

The optional HIPPI-6400-PH copper cable variant (see clause 16) uses a cable with 46 conductor pairs, 23 in each direction, and an overall shield. The maximum length is dependent upon the quality of the cable. The signals are ac coupled to the cable to accommodate some difference in the ground potential between the equipment.

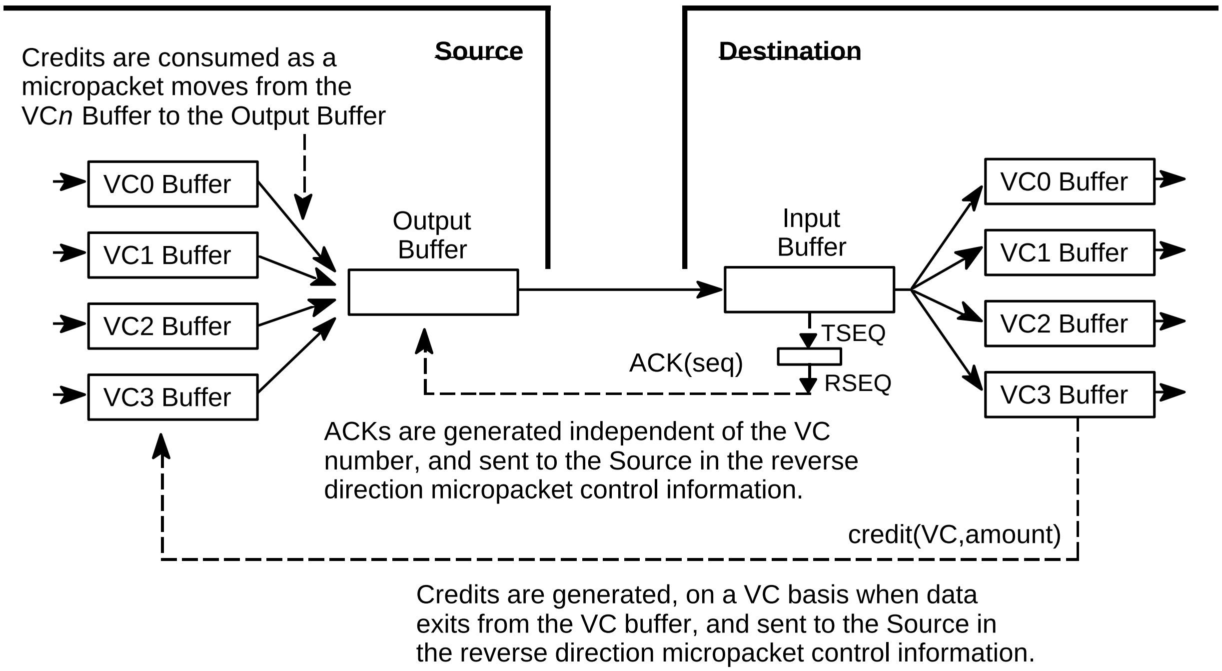


Figure 5 – Reverse direction control information

Table 1 – CRC coverages in a 128-byte Message

Micropacket number	Data Bytes DB00 – DB31 contents	ECRC coverage	LCRC coverage
1	Header	Header	Header, c00 – c47
2	Bytes 1 – 32	Header and Bytes 1 – 32	Bytes 1 – 32, c00 – c47
3	Bytes 33 – 64	Header and Bytes 1 – 64	Bytes 33 – 64, c00 – c47
4	Bytes 65 – 96	Header and Bytes 1 – 96	Bytes 65 – 96, c00 – c47
5	Bytes 97 – 128	Header and Bytes 1 – 128	Bytes 97 – 128, c00 – c47

5 Service interface

This clause specifies the services provided by HIPPI-6400-PH. The intent is to allow ULPs to operate correctly with this HIPPI-6400-PH. How many of the services described herein are chosen for a given implementation is up to that implementor; however, a set of HIPPI-6400-PH services must be supplied sufficient to satisfy the ULP(s) being used. The services as defined herein do not imply any particular implementation, or any interface.

Figure 6 shows the relationship of the HIPPI-6400-PH interfaces.

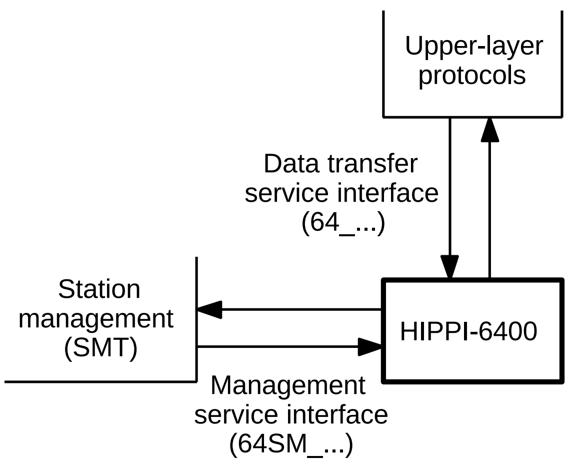


Figure 6 – HIPPI-6400-PH service interface

5.1 Service primitives

The primitives, in the context of the state transitions in clause 5, are declared required or optional. Additionally, parameters are either required, conditional, or optional. All of the primitives and parameters are considered as required except where explicitly stated otherwise.

HIPPI-6400-PH service primitives are of four types.

– *Request primitives* are issued by a service user to initiate a service provided by the HIPPI-6400-PH. In this standard, a second Request primitive of the same name shall not be issued until the Confirm for the first request is received.

–*Confirm primitives* are issued by the HIPPI-6400-PH to acknowledge a Request.

– *Indicate primitives* are issued by the HIPPI-6400-PH to notify the service user of a local event. This primitive is similar in nature to an unsolicited interrupt. Note that the local event may have been caused by a service Request. In this standard, a second Indicate primitive of the same name shall not be issued until the Response for the first Indicate is received.

– *Response primitives* are issued by a service user to acknowledge an Indicate.

5.2 Sequences of primitives

The order of execution of service primitives is not arbitrary. Logical and time sequence relationships exist for all described service primitives. Time sequence diagrams are used to illustrate a valid sequence. Other valid sequences may exist. The sequence of events between peer users across the user/provider interface is illustrated. In the time sequence diagrams the HIPPI-6400-PH users are depicted on either side of the vertical bars while the HIPPI-6400-PH acts as the service provider.

Open Issue – All of the rest of the service interface text in clause 5 is new and needs careful review. This is the first time it has seen the light of day and it will probably undergo significant change before it is finalized. The new text was marked with margin bars only, adding the highlights sometimes makes it harder to read.

5.3 Data transfer service primitives

These primitives, as shown in figure 7, shall be used to transfer ULP data from an Originating Source ULP to a Final Destination ULP. The ULP data shall be carried in a Message, with MAC and IEEE 802.2 LLC/SNAP Headers preceding the ULP payload data (see figures 4 and 13, and clause 7). The ULP data shall immediately follow the LLC/SNAP Header, i.e., the ULP payload data shall use the last eight bytes of the Header micropacket.

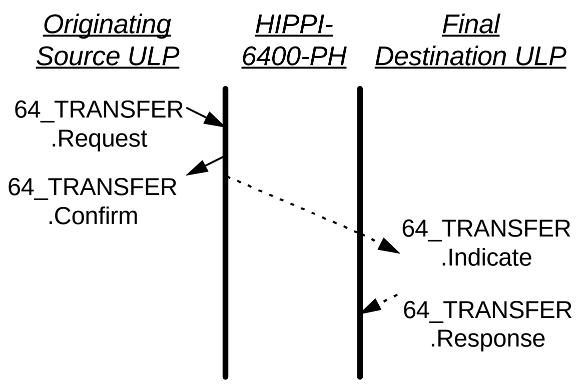


Figure 7 – Data transfer service primitives

5.3.1 64_TRANSFER.Request

Issued by the Originating Source's ULP to request a data transfer.

Semantics – 64_TRANSFER.Request (
D_ULA,
S_ULA,
VCn,
EtherType,
Length,
Data)

The D_ULA (Destination address) shall be placed directly in the MAC Header (see 7.1).

The S_ULA (Source address), if allowed by S_ULA_Allowed = true (see 5.4.1), shall be placed directly in the MAC Header (see 7.1). If S_ULA_Allowed = false, then the HIPPI-6400-PH entity shall use its native S_ULA address. Note that by allowing the ULP to specify the Source address, a server can use a "spoof" address, e.g., to provide a broadcast service. Whether the ULP is allowed to set the S_ULA or not is controlled by the local station management entity through the S_ULA_Allowed flag (see 5.4.1).

VCn shall be the Virtual Channel (see 6.2) that the message shall be sent on. If the Message size violates the Virtual Channel size limitations, then the Request shall be rejected.

EtherType, specifying the data type (see 7.2), shall be placed directly in the LLC/SNAP Header.

Length, shall specify the number of bytes of ULP payload data. Note that the length parameter in the MAC Header (see clause 7) M_len = Length + 8.

Data shall be the ULP payload data.

Issued – The Originating Source ULP issues this primitive to the HIPPI-6400-PH entity to request the transfer of the ULP payload data to the Final Destination.

Effect – The HIPPI-6400-PH entity shall accept the data for transmission and build the Message with the appropriate MAC and LLC/SNAP Headers. If the Message size violates the Virtual Channel limitations, then this transfer request shall be rejected (see 5.3.2); otherwise the Message shall be sent. If the Message does not end on a micropacket boundary then padding shall be provided (see clause 7).

5.3.2 64_TRANSFER.Confirm

This primitive acknowledges the 64_TRANSFER .Request from the Originating Source ULP.

Semantics – 64_TRANSFER.Confirm (Status)

Status shall be:

– Accept – The Message has been accepted for transmission.

– Reject – The Message violated the Virtual Channel size limitations (see 6.2), and has been rejected.

Issued – The HIPPI-6400-PH shall issue this primitive to the Originating Source ULP to acknowledge the 64_TRANSFER.Request.

Effect – Unspecified

5.3.3 64_TRANSFER.Indicate

This primitive indicates to the Final Destination ULP that a Message has been received from the Originating Source.

Semantics – 64_TRANSFER.Indicate (
D_ULA,
S_ULA,
EtherType,
Status,
Length,
Data)

The D_ULA shall be the value received in the MAC Header (see 7.1).

The S_ULA shall be the value received in the MAC Header (see 7.1).

EtherType shall be the value received in the LLC/SNAP Header (see 7.2).

Status denotes whether the data payload being delivered was received with errors. Status includes but is not limited to:

– ECRC errors (see 9.1.3);

– missing end of Message (see 9.2.3).

Length shall be the payload length as specified in the MAC Header, i.e., Length = M_len – 8.

Data shall be the ULP payload data with any pad removed.

Issued – The Final Destination HIPPI-6400-PH shall issue this primitive to the Final Destination ULP when a Message has been received.

Effect – Unspecified

5.3.5 64_TRANSFER.Response

This primitive acknowledges a 64_TRANSFER.In-dicate.

Semantics – 64_TRANSFER.Response

Issued – The Final Destination ULP issues this primitive to acknowledge the receipt of the 64_TRANSFER.Indicate.

Effect – The HIPPI-6400-PH Final Destination is enabled to issue another 64_TRANSFER.Indicate.

5.4 Control service primitives

These primitives, as shown in figure 8, may be used by the local station management (SMT) entity to set parameters and control the HIPPI-6400-PH interface.

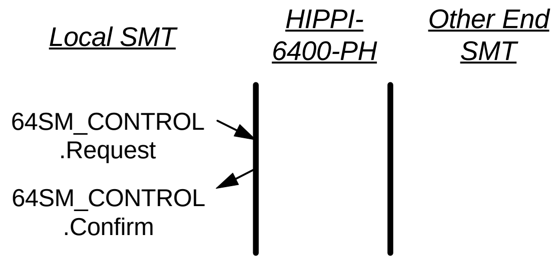


Figure 8 – Control service primitives

5.4.1 64SM_CONTROL.Request

Issued by the local SMT to set parameters, or otherwise control the HIPPI-6400-PH. Several functions are specified and others are left to specific implementations.

Semantics – 64SM_CONTROL.Request (
Command,
Command_Parameters)

Command specifies the function to be performed.

Command_Parameters are specific to each command.

The commands and parameters include but are not limited to:

– Set/reset S_ULA_Allowed flag (see 5.3.1)

– Set native S_ULA value (see 7.1)

– Set timeout values (see table 6)

– Link Reset (see 12.1)

– Initialize (see 12.2)

– Initialize logged events counters (see table 7)

Issued – The SMT issues this primitive to perform some control function.

Effect – The HIPPI-6400-PH shall perform the function specified.

5.4.2 64SM_CONTROL.Confirm

This primitive acknowledges the 64SM_CONTROL.Request from the SMT.

Semantics – 64SM_CONTROL.Confirm (Status)

Status reports the success or failure of the 64SM_CONTROL.Request commands.

5.5 Status service primitives

These primitives, as shown in figure 9, may be used to obtain status information from the local HIPPI-6400-PH entity.

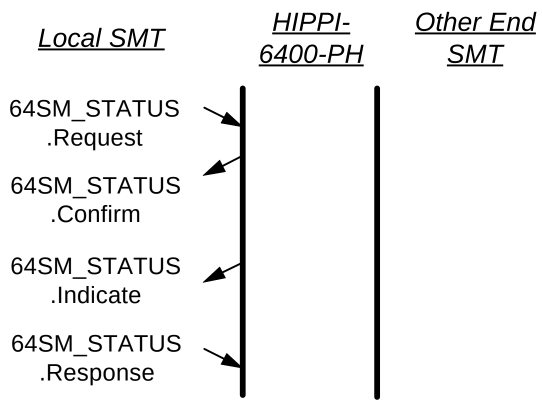


Figure 9 – Status service primitives

5.5.1 64SM_STATUS.Request

Issued by the local SMT to request a status report.

Semantics – 64SM_STATUS.Request

Issued – The local SMT issues this primitive when it wishes to obtain the status of the HIPPI-6400-PH entity. Note that an implementation may issue a blanket request for all of the status items, or for specific items; this is vendor specific.

Effect – The HIPPI-6400-PH entity shall respond with a 64SM_STATUS.Confirm.

5.5.2 64SM_STATUS.Confirm

This primitive replies to the previous 64SM_STATUS.Request with status information.

Semantics – 64SM_STATUS.Confirm (Status)

Status shall contain, but is not limited to:

– S_ULA_Allowed state (see 5.3.1)

– Native S_ULA value (see 7.1)

– Timeout values (see table 6)

– Logged events (see table 7)

– Activity monitor state (see 13.1)

– Link state, i.e., Normal, Resetting, Initializing, or Shutdown (see clause 12)

Issued – The HIPPI-6400-PH entity shall issue this primitive to the SMT in response to a 64SM_STATUS.Request.

Effect – Unspecified

5.5.3 64SM_STATUS.Indicate

This primitive informs the SMT entity that a major event has occurred that affects the operation of the HIPPI-6400-PH entity.

Semantics – 64SM_STATUS.Indicate

Issued – The HIPPI-6400-PH shall issue this primitive whenever a major event is detected. Major events shall include, but are not limited to:

– Activity monitor indication going false (see 13.1)

– Link going into Shutdown (see 13.2)

Effect – Unspecified. A normal response would be for the SMT entity to read the status and determine which event occurred.

5.5.5 64SM_STATUS.Response

This primitive acknowledges the 64SM_STATUS.Indicate.

Semantics – 64SM_STATUS.Response

Issued – The SMT entity issues this primitive to acknowledge receipt of the 64SM_STATUS.Indicate.

Effect – The HIPPI-6400-PH is allowed to issue another 64SM_STATUS.Indicate.

6 Micropacket contents

6.1 Bit and byte assignments

As shown in figure 3, each micropacket shall consist of 32 data bytes and 64 bits of control information. The data bytes shall be numbered DB00 - DB31. DB00 shall be transmitted first. The data bits in the micropacket shall be numbered d*xx.y* where *xx* is the byte number and *y* is the bit number in the byte.

The 64 bits of control information shall be numbered as bits c63 – c00. Control bit c00 shall be transmitted first. As shown in figure 3, a field with a numerical value shall have its most-significant bit in the highest numbered bit position.

The control information shall contain the following parameters located in the bits specified. The Source side of a link supplies all of the parameters, except for RSEQ, VCR and CR which come to the Source from its local Destination side. The VC parameter comes from the Originating Source. The TAIL, TYPE, and ECRC parameters normally come from the Originating Source, but may under error conditions come from an intermediate device (see 9.2.3 and 9.2.4).

VC (2 bits, c01–c00) – The Virtual Channel selector. (See 6.2.)

TYPE (4 bits, c05–c02) – Identifies the type of information within the micropacket. (See 6.3.)

TAIL (1 bit, c06) – TAIL = 1 identifies the last micropacket of a Message. TAIL = 0 means that more micropackets for this Message follow.

ERROR (1 bit, c07) – ERROR = 1 means that an unrecoverable error has been detected in the Message, do not check the ECRC. ERROR = 0 means that the Message is OK so far. (See 6.6.3 and 9.1.3.)

VCR (2 bits, c09–c08) – Virtual Channel number associated with credit addition. (See 6.5.)

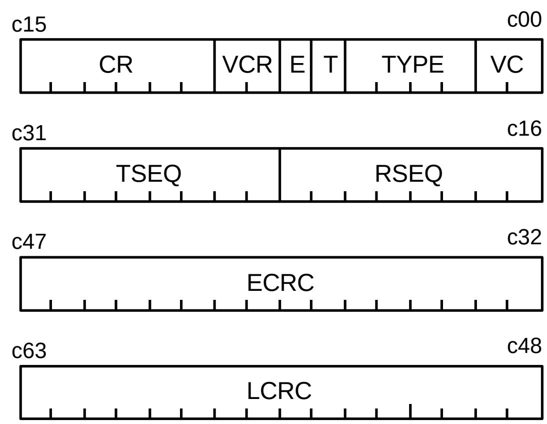
CR (6 bits, c15–c10) – Amount of credit to add to the Virtual Channel specified in VCR. (See 6.5.)

RSEQ (8 bits, c23–c16) – Sequence number associated with micropacket ACK indication. (See 6.4.)

TSEQ (8 bits, c31–c24) – Sequence number of transmitted micropacket. (See 6.4.)

ECRC (16 bits, c47–c32) – End-to-end checksum covering all of the data bytes up to this point in a Message, including those in the Header micropacket. (See 6.6.)

LCRC (16 bits, c63–c48) – Link level checksum covering the 32 data bytes, and the c00 through c47 control bits, in this micropacket. (See 6.6.)



Note – Transmission order is top to bottom, and right to left, in 4-bit groups, as shown in tables 3 and 4. The most-significant-bit of a parameter is at the left end of its field.

Figure 10 – Control bits summary

6.2 Virtual Channel (VC) selector

Four Virtual Channels shall be available in each direction on a link. Messages on the Virtual Channels shall be assigned as follows:

– VC0 = Messages with a maximum size of 68 data micropackets (2176 bytes) plus a Header micropacket.

– VC1 = Messages with a maximum size of 4100 data micropackets (~128 KBytes) plus a Header micropacket. Also carries Admin Request Messages.

– VC2 = Messages with a maximum size of 4100 data micropackets (~128 KBytes) plus a Header micropacket. Also carries Admin Response Messages.

– VC3 = Messages with a maximum size of 134,217,728 data micropackets (~4 GBytes) plus a Header micropacket. To avoid congestion, Originating Sources shall only initiate VC3 transfers to Final Destinations that have agreed to accept them, e.g., by using a Scheduled Transfer as specified in HIPPI-ST or by other unspecified means.

NOTE – The maximum Message size for VC0, VC1, and VC2 was picked to be an integral power of 2, plus up to 128 bytes for ULP header(s). For example, VC0's maximum Message size is 69 micropackets: one Header micropacket, four micropackets carrying 128 bytes of ULP header(s), and 64 micropackets carrying 2048 bytes of user payload.

6.3 Micropacket TYPEs

The 4-bit TYPE parameter shall indicate the contents of the micropacket.

Micropackets whose TYPE < x'8', or whose TYPE = x'A', are provided for control at the link level or for credit update. These micropackets are not loaded into any VC Buffer (see figure 5) at the Destination despite the VC field being transmitted as x'0'. As such, the Source need not have credit available for VC0 prior to sending these micropackets, and the Destination shall not generate additional VC0 credit as a result of having received these micropackets.

Only micropackets whose TYPE ≥ x'8' shall be retransmitted.

Undefined TYPE values are reserved for future use. Actions to be taken as a result of receiving an undefined TYPE are detailed in 9.1.4.

6.3.1 TYPE = link control micropackets

Control micropackets operate at the link level, do not carry any user data, acknowledgments, or credit update information. (See clause 12.) Control micropackets include:

– Reset (TYPE = x'2') – Sent to initiate a Link Reset operation. (See 12.1.)

– Reset_ACK (TYPE = x'3') – The receiving device has completed the Link Reset operation.

– Initialize (TYPE = x'4') – Sent to initiate an Initialization operation. (See 12.2.)

– Initialize_ACK (TYPE = x'5') – The receiving device has completed the Initialization operation.

6.3.2 TYPE = Null micropackets

Null micropackets (TYPE = x'7') are gap-fillers, and shall be used to keep the link active when there are no other micropackets to transmit. Null micropackets may carry ACK indications.

6.3.3 TYPE = Data micropackets

Data micropackets (TYPE = x'8') carry payload.

6.3.4 TYPE = Header micropackets

Header micropackets (TYPE = x'9') carry routing and control information.

6.3.5 TYPE = Credit-only micropackets

When credits are available, and there are no Data micropackets to send, then Credit-only micropackets (TYPE = x'A') are used to carry credit update information, and acknowledgments.

6.3.6 TYPE = Admin micropackets

Admin micropackets (TYPE = x'F') are used for support and initialization of HIPPI-6400 links, elements, and systems. Admin micropacket contents and uses are specified in HIPPI-6400-SC.

6.4 Sequence number parameters

The transmit sequence number (TSEQ) shall increment by one for each micropacket transmitted whose TYPE ≥ x'8'. TSEQ shall wrap from x'FE' to x'00'. The receive sequence number (RSEQ) shall be used to acknowledge (ACK) these micropackets. RSEQ shall equal the TSEQ of the most recent micropacket being acknowledged, or the latest TSEQ of a contiguous group of micropackets being acknowledged (see 9.3 and 8.2). TSEQ shall begin with the value = x'00' after a Link Reset. RSEQ = x'FF' indicates that no ACK indication is being transmitted (used while the link fills with micropackets after a Link Reset). TSEQ shall not overrun RSEQ, i.e., there shall be no more than 255 unacknowledged micropackets.

Table 2 – Micropacket contents summary

	Reset/ Initialize	Null	Credit-only	Header	Data	Admin
Data Bytes contents	0*	0*	0*	32 bytes of header information (see 7.1)	32 bytes of payload	Administrative information
VC	0*	0*	0*	any	any	Requests on VC1 Responses on VC2
TYPE (hex)	2,3,4,5	7	A	9	8	F
TAIL	1*	0*	0*	= 1 on last micropacket of Message	= 1 on last micropacket of Message	1
ERROR	0*	0*	0	= 1 if error	= 1 if error	= 1 if error
TSEQ	x'FF'	x'FF'	increments	increments	increments	increments
RSEQ	1*	ACK	ACK	ACK	ACK	ACK
VCR	0*	0*	any	any	any	any
CR	0*	0*	any	any	any	any
LCRC	single	single	single	single	single	single
ECRC	single	single	single	accumulating	accumulating	single

0* = transmit all bits of this field as 0's, a receiver must permit any value
1* = transmit all bits of this field as 1's, a receiver must permit any value
any = any data value as appropriate
single = this CRC is calculated and checked for this single micropacket
accumulating = ECRC as defined in 6.6.3

NOTES

1 The TSEQ and RSEQ parameters are independent of the Virtual Channel used to transmit the micropacket.

2 The TSEQ and RSEQ parameters are local to a specific link. For example, a micropacket that transverses more than one link will most likely have different TSEQ numbers on the different links.

3 The first micropacket with TYPE ≥ x'8' following a stomped micropacket (see 6.6.2.1) uses the same TSEQ value as in the stomped micropacket since that TSEQ value was not consumed.

The wrap at x'FE' shall be taken into account when processing ACK indications. For example, if the previous ACK indication had RSEQ = x'F7', and an ACK indication with RSEQ = x'03' is received, then the micropackets whose TSEQ value = x'F8' through x'FE', and from x'00' through x'03', are acknowledged and their memory may be reused by the Source.

6.5 Credit update parameters

The Destination shall insert the VCR and CR parameters in micropackets to inform the Source that CR number of micropacket buffers have been freed up for the VC indicated by VCR. The Source shall increase its Credit Counter for this Virtual Channel by the value in CR. The Source Credit Counter range shall be 255, and the number of outstanding credits shall be ≤ 255.

NOTES

1 The CR value is an incremental update value, not the number of buffers currently available in the Destination.

2 At 40 ns per micropacket, and 5 ns per meter of cable, each credit is equivalent to about 8 meters of cable. Hence, a Credit Count, and Destination buffer capacity per Virtual Channel, of 255 will support full bandwidth on a 1 km link when round

trip time is taken into account and Destination latency is low.

3 If the Destination does not send adequate credits then the Source may not be able to send on some VCs.

6.6 Check functions

6.6.1 Intended use of CRCs

Two 16-bit cyclic redundancy checks (CRCs) shall be used. The link CRC (LCRC) checks all of the data bytes and control bits in a single micropacket. The LCRC shall be generated by a link's Source, and checked by the same link's Destination, i.e., it is local to a link.

The end-to-end CRC (ECRC) checks all of the data bytes of a Message up to the end of the current micropacket of the Message, i.e., it may cover multiple micropackets. The ECRC shall be generated by the Originating Source, and should be passed unchanged through intermediate link-level devices. The ECRC shall be checked at each Destination in the path. See 9.1 for ECRC error operations.

While this standard covers the link level and host interface, other documents may require intermediate link-level devices to carry the ECRC across them, for example, across switches. Link-level devices, as described here, are devices that do not operate on the payload portion of data micropackets.

6.6.2 Link-level CRC (LCRC)

The link CRC (LCRC) shall cover all of the data bytes, and the control bits except for itself. The LCRC generator and checker shall be initialized to all ones (x'FFFF') for each micropacket.

The LCRC polynomial shall be:

$$x^{16} + x^{12} + x^{5} + 1$$

Figure 11 is an example serial implementation. The LCRC may be implemented in a parallel fashion rather than serial, but must produce the same results as the serial example. The c63 through c48 bits are the LCRC bits in the control word. The incoming data and control bits are exclusive OR'd with c48 to generate a *sum* value; the *sum* value is exclusive OR'd with selected control bits as they are shifted right once each bit period. The data and control bits shall be input to the generator in transmission sequence, i.e., 64 data bits, 16 control bits, 64 data bits, 16 control bits, etc. The sequence is d00.0, d00.1, d00.2, ...d00.7, d01.0...d01.7, ...d07.7, c00, c01, c02...c15, d08.0...d15.7, c16...c31, d16.0...d23.7, c32...c47, d24.0...d31.7. Refer to tables 3 and 4 for the transmission sequence. After passing all 304 input bits, c63-through-c48 contain the most-significant through least-significant bits of the LCRC.

At the destination, the LCRC check may be implemented by clocking the entire micropacket, including the LCRC parameter (c63..c48), into either a serial or parallel checker. In this case, a residue is available in the checker register after the last clock rather than a syndrome. If this check method is used, a residue of x'0000' indicates no errors, and x'06A9' indicates that a "stomp" code was received.

See 9.1.1 for details of a Destination's actions when checking the LCRC. See annex A.3 for the equations to generate the LCRC in a parallel fashion.

6.6.2.1 Stomp code at Source

A Source may decide during the course of transmitting a micropacket that it wishes to "nullify" that transmission. This shall be done by XORing a "stomp" code of x'874D' with the LCRC that it has calculated for the micropacket. The Source shall treat a "stomped" micropacket as if it never occurred, i.e., not save the "stomped" micropacket in the retransmit buffer, and not increment the TSEQ number since the TSEQ number was not consumed.

6.6.2.2 Stomp code at Destination

If the Destination detects a "stomp" code (see 6.6.2), then an LCRC error shall not be logged (see 9.1.1).

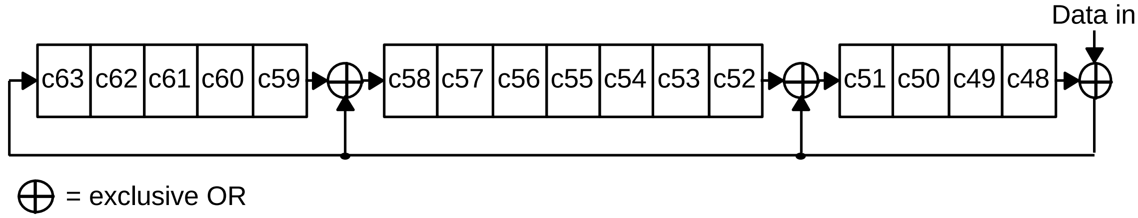


Figure 11 – LCRC implementation example

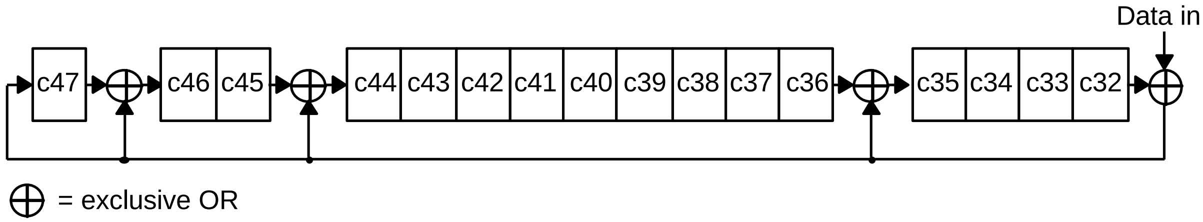


Figure 12 – ECRC implementation example

6.6.3 End-to-end CRC (ECRC)

The end-to-end CRC (ECRC) shall include only the micropacket's data bytes, not the control bits, in its calculation. The ECRC shall include all of a Message's date bytes up to this point in the Message, i.e., the data bytes in the Header micropacket and in all of the Data micropackets up to this point in the Message.

All Sources not generating the original ECRC shall check the ECRC prior to transmission, and if the ECRC is in error then set ERROR = 1 in this micropacket's control bits. An ECRC_Source_Error shall be logged for only the first occurrence of this error in a Message (see 14.1). This aids in error isolation and prevents endless retransmission loops.

The ECRC generator polynomial shall be:

$$x^{16} + x^{12} + x^{3} + x + 1$$

The ECRC is calculated and maintained independently for each VC. The ECRC checker and generator for a VC shall be initialized to all ones (x'FFFF') for each Message. Figure 12 is an example ECRC serial implementation. The ECRC may be implemented in a parallel fashion rather than serial, but must produce the same results as the serial example. The c47 through c32 bits are the ECRC bits in the control word. The incoming data bits are exclusive OR'd with c32 to generate a *sum* value; the *sum* value is exclusive OR'd with selected control bits as they are shifted right once each bit period. The data bits shall be input to the generator in transmission sequence, i.e., d00.0, d00.1, d00.2, ...d00.7, d01.0...d01.7, ...d31.7. Refer to tables 3 and 4 for the transmission sequence. After passing all 256 of the micropacket's data bits, c47-through-c32 contain the most-significant through least-significant bits of the ECRC for this micropacket. The ECRC value will normally be different for each micropacket of a Message since the ECRC accumulates as the Message progresses (see table 1).

See 9.1.3 for details of a Destination's actions when checking the ECRC. See annex A.4 for the equations to generate the ECRC in a 64-bit-wide fashion.

7 Message structure

As defined in 4.4, a Message is an ordered sequence of one or more micropackets which have the same VC, start with a Header micropacket (TYPE = Header), and have TAIL = 1 in the last micropacket. Each VC may only have a single Message in progress at any time. Since only complete micropackets are transmitted, a Message that is not an integral multiple of 32 bytes in length shall be padded with zeros in the last micropacket.

The Message header format is shown in figure 13 as a group of 32-bit words. The Media Access Control (MAC) Header, and LLC/SNAP header, shall reside in the first 24 bytes of all Header micropackets. If a parameter uses more than one byte, the lowest numbered byte is the most-significant byte. The last eight bytes of the Header may be used by other protocols, and are not defined in this standard.

7.1 MAC Header

The MAC header shall be included in all HIPPI-6400 Messages. The MAC header shall be in the first micropacket (TYPE = Header) of a Message, and shall contain:

D_ULA (48 bits, DB00-DB05) – The IEEE 48-bit ULA network address, as defined in ANSI/IEEE Std 802, identifying the payload's Final Destination. Figure 14 (following IEEE 802.1A canonical bit order, and HIPPI byte order) details the placement of the D_ULA.

S_ULA (48 bits, DB06-DB11) – The IEEE 48-bit ULA network address, as defined in ANSI/IEEE Std 802, identifying the payload's Originating Source. Figure 14 details the placement of the S_ULA.

M_len (32 bits, DB12-DB15) – The Message length, in bytes, following the M_len field, exclusive of any padding in the last micropacket.

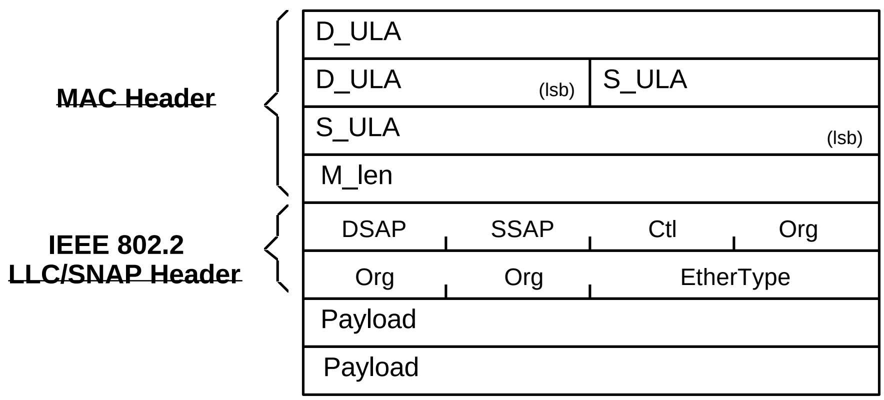


Figure 13 – Header micropacket contents

D_ULA Octet 0 U/L I/G	D_ULA Octet 1	D_ULA Octet 2	D_ULA Octet 3
D_ULA Octet 4	D_ULA Octet 5	S_ULA Octet 0 U/L I/G	S_ULA Octet 1
S_ULA Octet 2	S_ULA Octet 3	S_ULA Octet 4	S_ULA Octet 5

NOTE – U/L = 1 for Locally administered address, 0 for Universal; I/G = 1 for Group address, 0 for Individual

Figure 14 – Detailed ULA layout

7.2 LLC/SNAP header

The LLC/SNAP header, as defined in ISO/IEC 8802-2 (ANSI/IEEE Std. 802.2), shall be included in all Messages. The LLC/SNAP header shall be 64 bits (DB16-DB23) and shall immediately follow the MAC header in the first micropacket (Header micropacket). The values of the LLC/SNAP header subfields shall be: DSAP = x'AA' (i.e., SNAP), SSAP = x'AA' (i.e., SNAP), Ctl = x'03' (i.e., unnumbered packets), and the three Org = x'00' (i.e., generic packets). Codings of the EtherType field shall be as assigned in the current "Assigned Numbers" RFC, e.g., RFC 1700[1]. For the convenience of the reader, HIPPI-6400-specific EtherTypes are listed below:

x'8180' = HIPPI-FP as specified in ANSI X3.210. (See annex A.)

x'8181' = Scheduled Transfer, as specified in ANSI X3.xxx, HIPPI-ST.

x'8182' = Locally administered.

x'8183' = Reserved

7.3 Payload

The eight bytes following the LLC/SNAP header belong to the ULP using this Message. The payload bytes may be used to carry additional headers, parameters, or data.

8 Source specific operations

8.1 Credit update indications on Source side

Credit update indications from the remote end are received on the local Destination side, and passed to the local Source side, as shown in figure 5. A credit update shall increase the available credit, by the amount in the CR parameter, on the Virtual Channel whose number is the value in the VCR parameter.

If data is ready to be sent on a given VC, but credits are exhausted for this VC (i.e., credit = 0) for the duration of a timeout period, then the link is shut down (see 13.2), and a VC[0-3]_Credit_Timeout_Error logged. The default timeout value shall be 2 seconds (see 14.2).

If a credit update results in credit > 255 then the link shall be reset (see 12.1) and a VC[0-3]_Credit_Overflow_Error logged.

8.2 ACK indications on Source side

ACK indications (see 6.4 and 9.3) from the remote end are received on the local Destination side, and passed to the local Source side, as shown in figure 5. An ACK indication acknowledges all of the transmitted micropackets whose TSEQ ≤ RSEQ, i.e., the memory allocated to these micropackets may be re-used. RSEQ = x'FF', which may occur immediately after a Reset operation (see 9.3), shall be ignored.

The ACK indication timeout indicates that a TSEQ was transmitted, but not acknowledged for the length of time longer than the worst-case round trip time possible for an acknowledgment to occur. If the ACK indication timeout expires, the Source shall retransmit all micropackets, (see 8.4), that have not been acknowledged, and shall log an RSEQ_Missing_Error (see 14.1). The ACK indication timeout default value shall be 12 µs (see 14.2).

NOTE – The ACK indication timeout provides a recovery mechanism even in the event of lost RSEQ values due to link errors. Faster recovery may be possible with other schemes, e.g., NAKs, but the complexity required for the performance gain did not seem worth it, especially since errors should be infrequent.

If an illegal RSEQ value is received, the Source shall retransmit all micropackets, (see 8.4), that have not been acknowledged, and log an RSEQ_Out_Of_Range_Error (see 14.1). An illegal RSEQ is one that does not equal or fall between the last successfully received RSEQ and the highest transmitted but not acknowledged TSEQ.

[1] RFC (Request For Comment) documents are working standards documents from the TCP/IP internetworking community. Copies of these documents are available from numerous electronic sources (e.g., http://www.ietf.org) or by writing to IETF Secretariat, c/o Corporation for National Research Initiatives, 1895 Preston White Drive, Suite 100 Reston, VA 20191-5434, USA.

8.3 ACKs and credit updates to remote end

The local Destination side sends ACK indications and credit update information to the remote end by first queuing them to the local Source side, as shown in figure 5. The Source side shall transmit this information in micropackets using the appropriate control bits. Since the ACK indications and credit update information do not share their fields with any other parameters they can be sent with every micropacket.

The local Destination may queue multiple ACK indication RSEQ parameters before one is transmitted by the local Source end. The RSEQ parameter should be over-written so that the ACK indication Message transmitted uses the latest value of RSEQ.

8.4 Micropacket retransmission

A retransmission sequence, as triggered by the error conditions defined in 8.2, shall consist of two consecutive training sequences (see 11) followed by retransmission of all of the unacknowledged micropackets in the Output Buffer (see figure 5).

Multiple retransmissions may be required in the event of poor link quality. The link shall be shut down (see 13.2), and a Retransmissions error logged (see 14.1), if successful operation is not achieved after a number of successive retransmissions of the same data. The default number is two, and it shall be programmable to other values, including 1 and 4. The mechanisms and procedures used to set values, different from the default value, are outside the scope of this standard.

NOTE – This number of allowed consecutive retransmissions may need to be larger to accommodate lengthy noise hits or to provide equivalent noise immunity when smaller ACK indication timeout values (see 8.2) are chosen for short cables.

Upon retransmission, the following parameters, from the original micropacket, shall have the same value in the retransmitted micropacket.

- VC
- TYPE
- TAIL
- ERROR
- TSEQ
- VCR
- CR
- ECRC

The following parameters may change as a micropacket is retransmitted.

- RSEQ
- LCRC

9 Destination specific operations

9.1 Link level processing

The Destination shall process received micropackets in the order of the following subclauses. The unnumbered items within each subclause may be checked in any order. Note that no acknowledgment (i.e., with RSEQ) shall be given for a micropacket that is discarded.

9.1.1 Check received LCRC

- If LCRC syndrome = x'874D' (stomp code) then the Destination shall discard the micropacket, and not log an error.

- If LCRC syndrome ≠ x'0000', and ≠ x'874D' (stomp code), then the Destination shall discard the micropacket and log an LCRC_Error.

9.1.2 Check received TSEQ

If no errors were detected in 9.1.1, and TSEQ ≠ x'FF', then the following checks shall be made. TYPE < x'8' is an error. TYPE ≥ x'8', and TSEQ is not one greater than the last non-x'FF', non-stomped, TSEQ received, is also an error. In either case, the micropacket shall be discarded. Additionally, a TSEQ_Error shall be logged unless no micropackets have been accepted since the last TSEQ_Error was logged.

9.1.3 Check received ECRC

If no errors were detected in 9.1.1 or 9.1.2, then the following checks shall be made.

– If ERROR = 0 and the ECRC syndrome ≠ x'0000', then the Destination shall discard the micropacket and log an ECRC_Error.

– If ERROR = 1 and the ECRC syndrome ≠ x'0000', then the Destination shall process the micropacket as if the ECRC were correct (unless this is the Final Destination in which case an error shall be signalled to the ULP).

9.1.4 Undefined TYPE

If TYPE = undefined and is in the range of x'0' - x'7' then the Destination shall treat the micropacket as a Null micropacket. If TYPE = undefined and in the range of x'8' - x'F' then intermediate Destinations shall treat the micropacket as a Data micropacket. Treatment by a Final Destination is not specified by this standard. For any undefined TYPE value, a VC[0-3]_Undefined_TYPE_Error shall be logged and the most recent offending TYPE value stored in Undefined_TYPE_Value.

NOTE – The actions applied to Undefined TYPEs are intended to allow for future use of the Undefined TYPE values.

9.2 Check for Message protocol errors

Message protocol error checking (at the Destination) shall be done on micropackets that have not been discarded in 9.1 and its subclauses. Since a Message is restricted to a single Virtual Channel, all Message protocol checking shall be applied to each Virtual Channel independently. Credit-only (TYPE = x'A') micropackets shall be ignored for the purposes of Message protocol checking. Otherwise, micropackets shall be checked in the order received on each Virtual Circuit.

9.2.1 Admin missing TAIL bit

If TYPE = Admin, and Tail = 0, then the Destination shall forward the Admin micropacket with ERROR = 1, and TAIL = 1. A VC[1-2]_Admin_Tail_Error shall be logged.

9.2.2 Missing start of Message

If a Message is missing the Header micropacket (i.e., a micropacket with TYPE = Data or undefined is received following a micropacket with TAIL = 1, or a Link Reset operation) then the Destination shall process the micropackets on this VC until a micropacket with TYPE = Header or Admin is received. This processing for the micropackets shall consist of discarding the data bytes; their control information shall be treated normally and RSEQs shall be generated. Subsequent Header or Admin micropackets shall be treated normally. The Destination shall log a VC[0-3]_Missing_Start_of_Message_Error for each discarded Message; not log an error for each discarded micropacket.

9.2.3 Missing end of Message

If the end of a Message is missing (i.e., TYPE = Header or Admin following a Data, Header, or undefined TYPE ≥ x'8' micropacket with TAIL = 0) then the Destination shall fabricate an end of Message micropacket (Data Bytes = x'00', VC = as received, TYPE = Data, TAIL = 1, ERROR = 1, other parameters as appropriate). The Destination shall insert the fabricated micropacket into the VC stream, and shall log a VC[0-3]_Missing_End_of_Message_Error. The Header or Admin micropacket shall be treated normally.

9.2.4 Stall timeout

If a Message is in progress on a VC, that VC's buffer is empty, and no Data micropackets have been received within the Stall timeout period, then the Destination shall fabricate an end of Message micropacket (Data Bytes = x'00', VC = as received, TYPE = Data, TAIL = 1, ERROR = 1, other parameters as appropriate). The Destination shall insert the fabricated micropacket into the VC stream, and shall log a VC[0-3]_Stall_Timeout_Error. This action flushes the Message in progress. The default value of the Stall timeout shall be 2 ms (see 14.2).

NOTE – Implementors are cautioned that the Stall timeout may be triggered by a slow Source host. If slow hosts are expected, then the Stall timeout value may be set to a larger value to avoid inadvertent actions.

9.2.5 No errors detected

If no errors are detected, and TYPE = Header or Data, the micropacket shall be acknowledged and delivered to the Virtual Channel buffer designated by the VC parameter.

If no errors are detected, and TYPE ≠ Header or Data, the micropacket shall be processed by the Destination.

9.3 Generating ACKs

The Destination acknowledges correctly received micropackets by using the RSEQ parameter of micropackets flowing in the reverse direction. Multiple micropackets may be acknowledged with a single RSEQ (e.g., if micropackets with TSEQ = 0,1...7 are received, transmitting RSEQ = 5 acknowledges micropackets 0,1...5, but not 6 and 7). Only micropackets that are not discarded due to errors (see 9.1) and whose TYPE value is in the range x'8' – x'F' shall be acknowledged. If the Destination does not have a new value of RSEQ to send, it shall repeat the last RSEQ value.

Once an error is detected that causes a micropacket not to be acknowledged, the Destination shall not change the RSEQ value until correctly receiving a micropacket with TSEQ = RSEQ + 1 (the retransmission of the micropacket that was in error). Hence, an error will result in a given RSEQ value being continually sent, and the Source timing out waiting for the expected RSEQ value (i.e., RSEQ > last RSEQ).

The Destination shall use RSEQ = x'FF' after a Link Reset or Initialize operation until it has received the micropacket with TSEQ = x'00'.

10 Signal line encoding

10.1 Signal line bit assignments

The data bytes and control bits shall be transmitted on the signal lines specified in table 3 for a 16-bit wide interface, and as specified in table 4 for an 8-bit wide interface. Nomenclature for the data and control bits is detailed in figure 3. Data signal lines are labeled capital D and a two-digit number, e.g., D00. Control signal lines are labeled capital C and a one-digit number, e.g., C0. The horizontal rows correspond to logical clock ticks. They are grouped in fours, corresponding to the 4b/5b coding.

Table 3 – Signal line bit assignments in a 16-bit system

	Signal lines																			
bit	**C3**	**C2**	**C1**	**C0**	**D 15**	**D 14**	**D 13**	**D 12**	**D 11**	**D 10**	**D 09**	**D 08**	**D 07**	**D 06**	**D 05**	**D 04**	**D 03**	**D 02**	**D 01**	**D 00**
a	12	08	04	00	07.4	07.0	06.4	06.0	05.4	05.0	04.4	04.0	03.4	03.0	02.4	02.0	01.4	01.0	00.4	00.0
b	13	09	05	01	07.5	07.1	06.5	06.1	05.5	05.1	04.5	04.1	03.5	03.1	02.5	02.1	01.5	01.1	00.5	00.1
c	14	10	06	02	07.6	07.2	06.6	06.2	05.6	05.2	04.6	04.2	03.6	03.2	02.6	02.2	01.6	01.2	00.6	00.2
d	15	11	07	03	07.7	07.3	06.7	06.3	05.7	05.3	04.7	04.3	03.7	03.3	02.7	02.3	01.7	01.3	00.7	00.3
a	28	24	20	16	15.4	15.0	14.4	14.0	13.4	13.0	12.4	12.0	11.4	11.0	10.4	10.0	09.4	09.0	08.4	08.0
b	29	25	21	17	15.5	15.1	14.5	14.1	13.5	13.1	12.5	12.1	11.5	11.1	10.5	10.1	09.5	09.1	08.5	08.1
c	30	26	22	18	15.6	15.2	14.6	14.2	13.6	13.2	12.6	12.2	11.6	11.2	10.6	10.2	09.6	09.2	08.6	08.2
d	31	27	23	19	15.7	15.3	14.7	14.3	13.7	13.3	12.7	12.3	11.7	11.3	10.7	10.3	09.7	09.3	08.7	08.3
a	44	40	36	32	23.4	23.0	22.4	22.0	21.4	21.0	20.4	20.0	19.4	19.0	18.4	18.0	17.4	17.0	16.4	16.0
b	45	41	37	33	23.5	23.1	22.5	22.1	21.5	21.1	20.5	20.1	19.5	19.1	18.5	18.1	17.5	17.1	16.5	16.1
c	46	42	38	34	23.6	23.2	22.6	22.2	21.6	21.2	20.6	20.2	19.6	19.2	18.6	18.2	17.6	17.2	16.6	16.2
d	47	43	39	35	23.7	23.3	22.7	22.3	21.7	21.3	20.7	20.3	19.7	19.3	18.7	18.3	17.7	17.3	16.7	16.3
a	60	56	52	48	31.4	31.0	30.4	30.0	29.4	29.0	28.4	28.0	27.4	27.0	26.4	26.0	25.4	25.0	24.4	24.0
b	61	57	53	49	31.5	31.1	30.5	30.1	29.5	29.1	28.5	28.1	27.5	27.1	26.5	26.1	25.5	25.1	24.5	24.1
c	62	58	54	50	31.6	31.2	30.6	30.2	29.6	29.2	28.6	28.2	27.6	27.2	26.6	26.2	25.6	25.2	24.6	24.2
d	63	59	55	51	31.7	31.3	30.7	30.3	29.7	29.3	28.7	28.3	27.7	27.3	26.7	26.3	25.7	25.3	24.7	24.3

NOTES

1 The two-digit numbers in the C*n* columns are the control bits, c*nn.*

2 The three-digit numbers in the D*nn* columns are the data bits, d*xx.y,* where *xx* is the byte number and *y* is the bit number in the byte.

3 The 4-bit groups in a column are transmitted on the associated signal line, top group first, bottom group last.

4 The four-bit groups in a column denote 4-bit code groups (dcba) for encoding/decoding to/from the 5-bit transmission codes (zyTxw) specified in table 5. A 5-bit group code (wxTyz) is transmitted over one signal line, e.g., D00.

Table 4 – Signal line bit assignments in an 8-bit system

	Signal lines									
bit	**C1**	**C0**	**D 07**	**D 06**	**D 05**	**D 04**	**D 03**	**D 02**	**D 01**	**D 00**
a	08	00	07.0	06.0	05.0	04.0	03.0	02.0	01.0	00.0
b	09	01	07.1	06.1	05.1	04.1	03.1	02.1	01.1	00.1
c	10	02	07.2	06.2	05.2	04.2	03.2	02.2	01.2	00.2
d	11	03	07.3	06.3	05.3	04.3	03.3	02.3	01.3	00.3
a	12	04	07.4	06.4	05.4	04.4	03.4	02.4	01.4	00.4
b	13	05	07.5	06.5	05.5	04.5	03.5	02.5	01.5	00.5
c	14	06	07.6	06.6	05.6	04.6	03.6	02.6	01.6	00.6
d	15	07	07.7	06.7	05.7	04.7	03.7	02.7	01.7	00.7
a	24	16	15.0	14.0	13.0	12.0	11.0	10.0	09.0	08.0
b	25	17	15.1	14.1	13.1	12.1	11.1	10.1	09.1	08.1
c	26	18	15.2	14.2	13.2	12.2	11.2	10.2	09.2	08.2
d	27	19	15.3	14.3	13.3	12.3	11.3	10.3	09.3	08.3
a	28	20	15.4	14.4	13.4	12.4	11.4	10.4	09.4	08.4
b	29	21	15.5	14.5	13.5	12.5	11.5	10.5	09.5	08.5
c	30	22	15.6	14.6	13.6	12.6	11.6	10.6	09.6	08.6
d	31	23	15.7	14.7	13.7	12.7	11.7	10.7	09.7	08.7
a	40	32	23.0	22.0	21.0	20.0	19.0	18.0	17.0	16.0
b	41	33	23.1	22.1	21.1	20.1	19.1	18.1	17.1	16.1
c	42	34	23.2	22.2	21.2	20.2	19.2	18.2	17.2	16.2
d	43	35	23.3	22.3	21.3	20.3	19.3	18.3	17.3	16.3
a	44	36	23.4	22.4	21.4	20.4	19.4	18.4	17.4	16.4
b	45	37	23.5	22.5	21.5	20.5	19.5	18.5	17.5	16.5
c	46	38	23.6	22.6	21.6	20.6	19.6	18.6	17.6	16.6
d	47	39	23.7	22.7	21.7	20.7	19.7	18.7	17.7	16.7
a	56	48	31.0	30.0	29.0	28.0	27.0	26.0	25.0	24.0
b	57	49	31.1	30.1	29.1	28.1	27.1	26.1	25.1	24.1
c	58	50	31.2	30.2	29.2	28.2	27.2	26.2	25.2	24.2
d	59	51	31.3	30.3	29.3	28.3	27.3	26.3	25.3	24.3
a	60	52	31.4	30.4	29.4	28.4	27.4	26.4	25.4	24.4
b	61	53	31.5	30.5	29.5	28.5	27.5	26.5	25.5	24.5
c	62	54	31.6	30.6	29.6	28.6	27.6	26.6	25.6	24.6
d	63	55	31.7	30.7	29.7	28.7	27.7	26.7	25.7	24.7

NOTES:

1 The two-digit numbers in the C*n* columns are the control bits, c*nn*.

2 The three-digit numbers in the D*nn* columns are the data bits, d*xx*.*y*, where *xx* is the byte number and *y* is the bit number in the byte.

3 The 4-bit groups in a column are transmitted on the associated signal line, top group first, bottom group last.

4 The four-bit groups in a column denote 4-bit code groups (dcba) for encoding/decoding to/from the 5-bit transmission codes (zyTxw) specified in table 5. A 5-bit code group (wxTyz) is transmitted over one signal line, e.g., D00.

10.2 Source-side encoding for dc balance

The transmitted signals shall be encoded to achieve dc balance on each signal line. Table 5 specifies the 5-bit signal line codes (zyTxw) corresponding to the 4-bit input codes (dcba) from tables 3 and 4. For example, on signal line D00, the first dcba 4-bit code consists of bits d00.0, d00.1, d00.2, and d00.3. See annex A.1 for an example circuit.

For each signal line, a running count, called the Disparity Count, shall be kept of all the ones and zeros transmitted on that line since the link was reset. The Disparity Count shall be incremented for each one transmitted, and decremented for each zero transmitted.

The appropriate 5-bit code value from table 5, based on the current value of the Disparity Count, shall be transmitted in the sequence, w,x,T,y,z. (T = true/complement bit) For example in the right column of tables 3 and 4, if:

a = d00.0 = 1 (least-significant bit)
b = d00.1 = 0
c = d00.2 = 0
d = d00.3 = 0
and Disparity Count = +1 before encoding,

then, based on the third column second row in table 5, transmit on D00:

w = 1 (transmitted first)
x = 0
T = 1
y = 0
z = 0

Disparity Count = 0 after encoding.

NOTES

1 The range for the Disparity Count at the 5-bit boundaries is from +4 to -5. The range for the Disparity Count is from +6 to -7.

2 The Disparity Count may also be updated by adding or subtracting the value of Delta Disparity shown in table 5. Add Delta Disparity if Disparity Count < 0; subtract if ≥ 0.

3 The 5-bit code is derived by inserting a 1 in the middle of the 4-bit code, and then transmitting either the true or complement value of the resultant 5-bit quantity.

4. The maximum run length, i.e., the longest string of continuous 1s or 0s, is 11. The string of 4-bit code points creating the maximum run length is x'EFC'. Start with Disparity Count = +3 or +4 for a string of 11 zeros. Start with Disparity Count = -4 or -5 for a string of 11 ones.

The data and control signal lines shall be synchronized with the CLOCK, CLOCK_2, and FRAME signals as shown in figures 15 and 17. Figures 15 through 18 are read left to right, i.e., events on the left occur before those on the right. In figures 15 and 17, the CLOCK_2 signal is deliberately shown skewed in relation to the CLOCK signal, although in actual implementation it may not be skewed (see 15.1).

Table 5 – 4b/5b line coding

4-bit code dcba	5-bit code when Disparity < 0 zyTxw	5-bit code when Disparity ≥ 0 zyTxw	Delta Disparity
0000	11011	00100	3
0001	11010	00101	1
0010	11001	00110	1
0011	00111	11000	1
0100	10011	01100	1
0101	01101	10010	1
0110	01110	10001	1
0111	01111	10000	3
1000	01011	10100	1
1001	10101	01010	1
1010	10110	01001	1
1011	10111	01000	3
1100	11100	00011	1
1101	11101	00010	3
1110	11110	00001	3
1111	11111	00000	5

10.3 Destination-side decoding

The received signals shall each be decoded in groups of five bits according to table 5.

NOTES

1 Decoding can be implemented by examining the middle bit of the 5-bit code; if 1 then use the outer bits uncomplemented, if 0 then complement before use.

2 There are no illegal 5-bit codes.

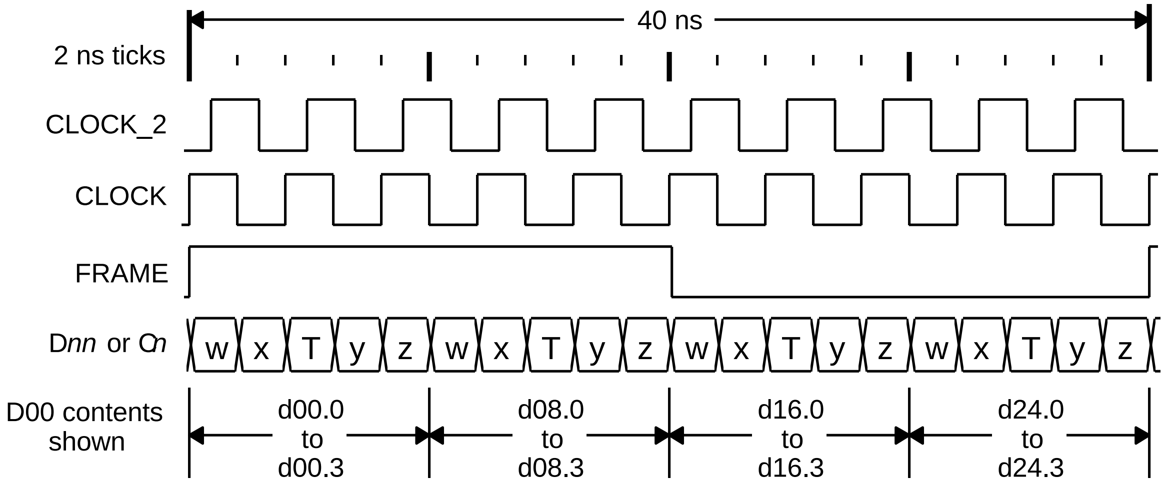


Figure 15 – 16-bit system micropacket

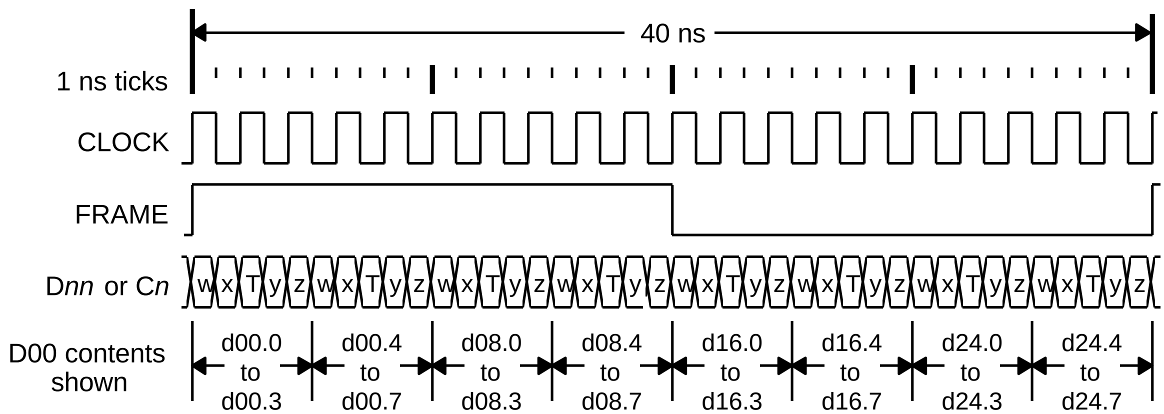


Figure 16 – 8-bit system micropacket

10.4 FRAME signal

The FRAME signal transitions shall be as shown in figures 15 through 18. As shown in figures 17 and 18, the start of a training sequence (40 ns long) shall be signaled by a 10101 FRAME signal pattern.

As shown in figures 15 and 16, a 0 to 1 transition on the FRAME signal shall signal the beginning of a micropacket, unless the transition is part of a training sequence. In a micropacket, the FRAME signal shall remain = 1 for the first half of the micropacket (20 ns), and shall = 0 for the last half (20 ns).

11 Skew compensation

11.1 Training sequences

The Destination shall compensate for up to 10 ns of skew among the signals. Skew is defined as the time between the earliest and latest signal arrival at the Destination. Training sequences (see figures 17 and 18) shall be used to measure the skew, and perform dynamic skew adjustments. A FRAME signal pattern of 10101, as specified in 10.4, shall be used to identify a training sequence.

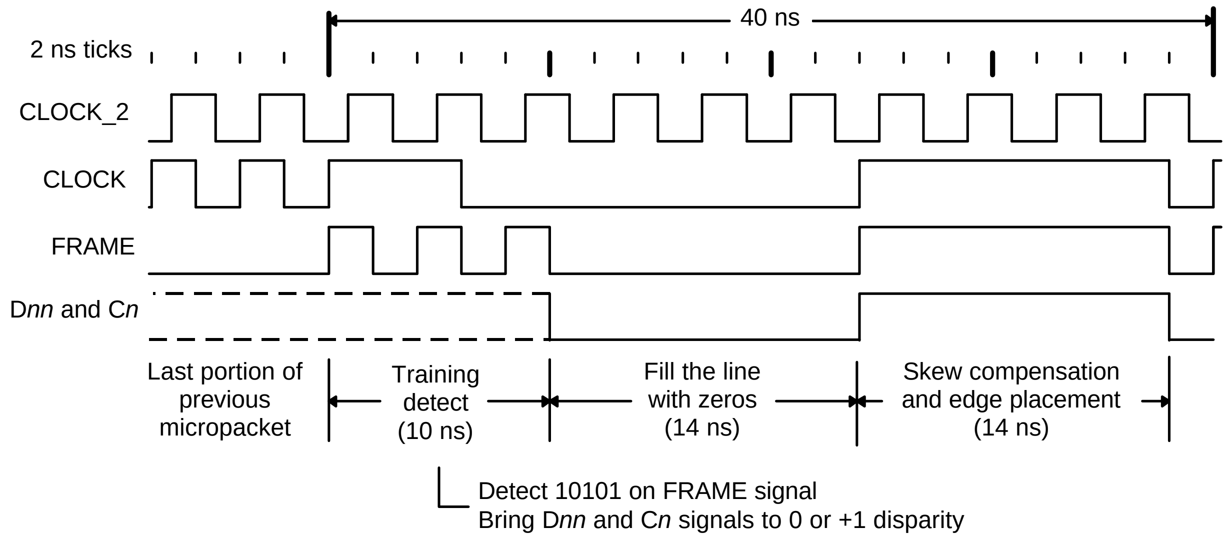


Figure 17 – 16-bit system training sequence

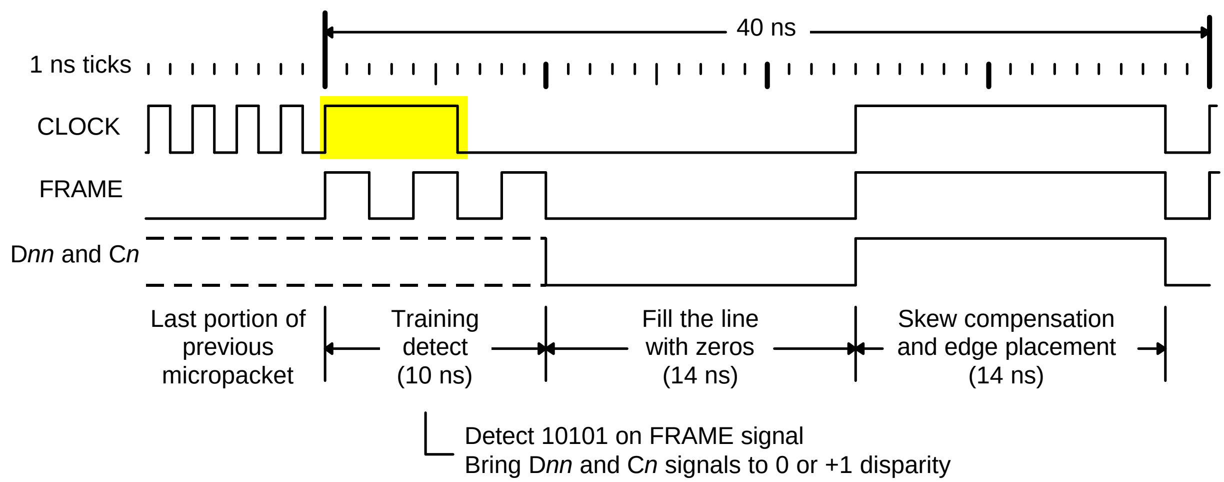


Figure 18 – 8-bit system training sequence

A single training sequence shall be inserted by the Source at least every 10 µs to adjust the dynamic skew, and also to compensate for CLOCK frequency differences between the Source and Destination (see annex A.2) During the first portion of a training sequence the Source shall insert appropriate Data and Control bits to drive the Disparity Count (see 10.2) on those signal lines to 0 or +1. The Disparity Count shall be set to zero at the end of the training sequence.

11.2 Training sequence errors

If the link fails to successfully train and complete its reset/initialize within a default Dead-man time of 100 ms (see 14.2) from any reset/initialize, then the link shall be reset (see 12.1) and a Reset_Initialize_Error shall be logged.

If the periodic retraining sequences fail to successfully re-train for any contiguous 100 ms Dead-man period (see 14.2) after the link had been healthy, then the link shall be reset (see 12.1) and a Skew_Retraining_Error shall be logged.

12 Link Reset and Initialization

Two levels of initialization are specified, Link Reset and Initialize. Link Reset affects the local link only; Initialize may be propagated to other links. Link Reset and Initialize operations are diagrammed in figure 19. A system power-on transition shall trigger either a Link Reset or Initialize sequence, the choice is implementation and system dependent, and is independent of the Hold-off timer (see 12.3). Link shutdown (see 13.2) occurs when the link is fatally flawed.

12.1 Link Reset

Link Reset affects the local link only, i.e., it is not propagated to other links. When the activity monitor indication = true (see 13.1), Power-on = true, and a Link Reset sequence is not currently in progress, then a Link Reset sequence may be triggered by the local administrator, and shall be triggered by:

– a RESET Admin command (see HIPPI-6400-SC);

– the Dead-man timer expires during an Initialize sequence (see 11.2 and figure 19);

– credit overflow (see 8.1);

– or receiving a micropacket with TYPE = Reset (see 6.3.1).

A Link Reset shall also be triggered by the activity monitor indication going from false to true (see 13.1).

During a Link Reset sequence:

– The receiver shall discard all micropackets except those with TYPE = Reset, Reset_ACK, Initialize, or Initialize_ACK, (i.e., with TYPE = x'2' – x'5').

– The error logging specified in clause 9 shall not occur.

– The error counts accessible by Admin operations (see table 7) shall be initialized if the Link Reset was triggered by a system power-on transition, otherwise they shall not be modified.

Exit from a Link Reset sequence occurs when a TYPE = Reset_ACK micropacket is received from the other end of the link, indicating that both ends of the link have completed the Reset sequence. At exit:

– all of the VC input and output buffers shall be emptied;

– credit for all of the VCs shall be set to zero;

– TSEQ shall be reset to x'00'; RSEQ shall be set to x'FF';

– the dynamic skew compensation circuitry adjusted, and micropackets being received correctly;

– and the Disparity Count shall be accurate.

12.2 Initialize

Initialize sequences may be propagated to other entities. When the activity monitor indication = true (see 13.1), Power-on = true, an Initialize sequence:

– may be triggered by the local administrator;

– shall be triggered by receiving a micropacket with TYPE = Initialize (see 6.3.1), and the Hold-off timer is expired or not running (see 12.3), and not currently doing an Initialize sequence (see figure 19).

During an Initialize sequence, the same actions as specified in 12.1 for a Reset sequence shall occur. In addition, an Initialize indication shall be passed to the local administrator for possible propagation to other entities.

The normal exit from an Initialize sequence occurs when a TYPE = Initialize_ACK micropacket is received from the other end of the link, indicating that both ends of the link have executed the Initialize sequence. If the link does not complete the Initialize sequence, i.e., receive a TYPE = Initialize_ACK micropacket, within the Dead-man timer period (see 11.2 and table 6), then the Initialize sequence shall be terminated and a Reset sequence started (see figure 19).

NOTE – The rationale for terminating an Initialize sequence within a limited time is to prevent stale Initialize sequences from propagating through a HIPPI-6400 network, i.e., Reset affects only the local link while Initialize may be propagated through multiple links.

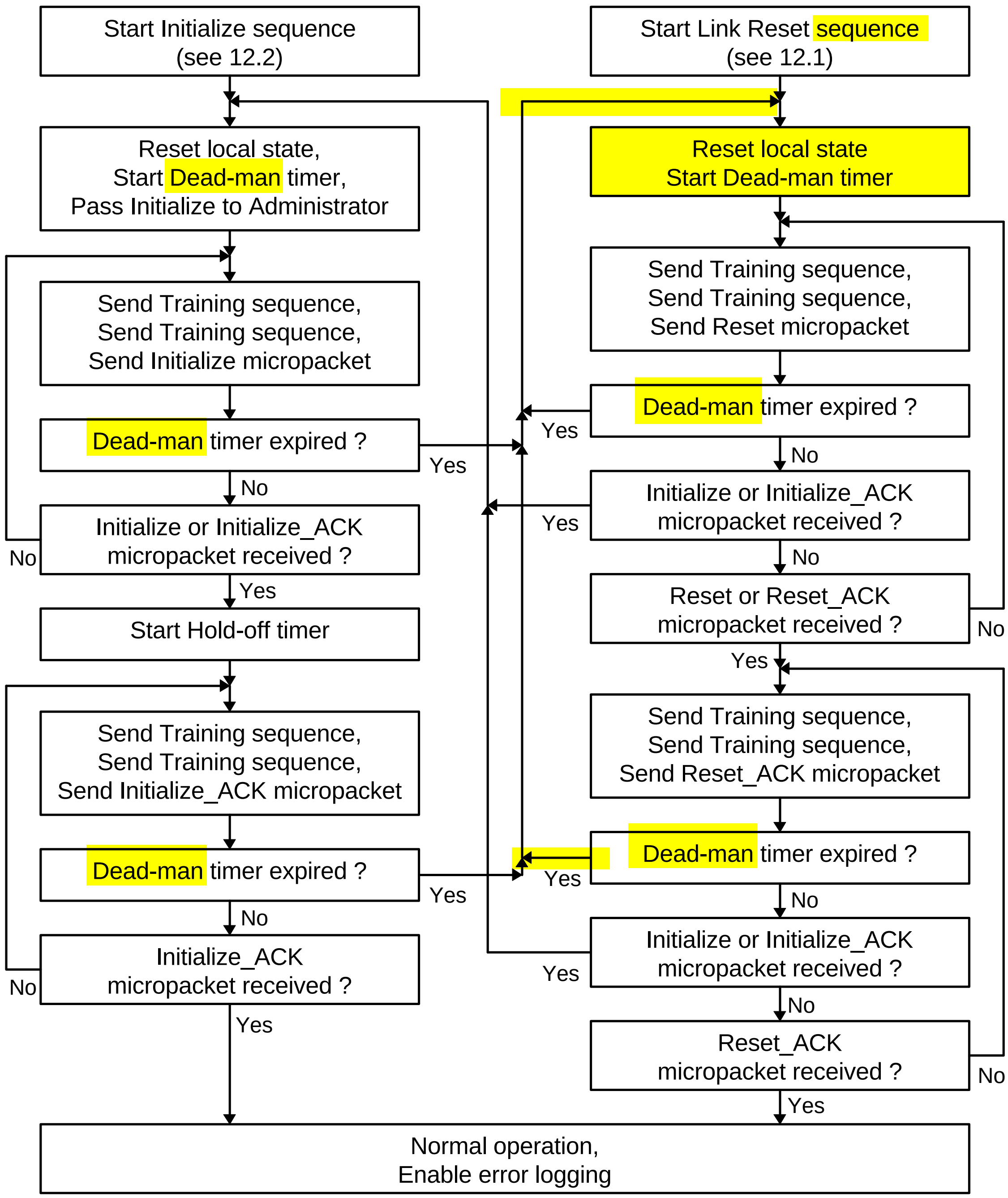


Figure 19 – Initialize and Link Reset operations

Upon exit from an Initialize sequence, the local state shall be reset the same as specified in 12.1, and all of the timeout timers (see table 6), except for the Hold-off timer, shall be initialized.

12.3 Hold-off timer

A Hold-off timer shall be used to prevent infinite Initialize oscillations among connected devices. The Hold-off timer shall be started by the first receipt of a TYPE = Initialize, or Initialize_ACK, micropacket. Until expired, the Hold-off timer shall be used to prohibit incoming TYPE = Reset or Initialize micropackets from starting a Reset or Initialize sequence. The default value for the Hold-off timer shall be 10 seconds (see 14.2).

13 Link activity monitoring and shutdown

13.1 Activity monitoring

The activity monitor shall be used to verify that the interconnecting media is present, and signals are being passed over the link.

The data (D*nn*), control (C*n*), FRAME, and CLOCK signals are affected by the activity monitor indication. When the activity monitor indication = true,

– The outputs of the transmitting circuit(s) shall be as specified in 15.1 or 16.1.

– The circuit(s) receiving the signals shall be enabled.

When the activity monitor indication = false, the transmitting and receiving circuits may be disabled to prevent damage to the interface or personnel (see 15.3 and 16.3).

The activity monitor shall be tolerant of the received signal, riding through minor signal aberrations during the Activity_Monitor timeout. For example, the activity monitor indication shall change after the detected signal has been stable in its new state (i.e., provide hysteresis) for at least the Activity_Monitor timeout period. The default value of the Activity_Monitor timeout shall be 1 ms (see 14.2).

13.2 Link shutdown

A link shutdown shall be triggered by:

– The number of times retransmission occurs exceeds some limit (see 8.4).

– The Source has been unable to transmit due to lack of credit (see 8.1).

– A Destination receives micropackets for a VC whose VC Buffer (see figure 5) is full. In this case, a VC[0-3]_RX_VC_Buffer_Overflow error logged shall be logged.

During a link shutdown:

– The local transmitter shall send continuous Null micropackets (with training sequences at appropriate intervals).

– The local transmitter shall deassert VC flow control to upstream receiver(s), i.e., micropackets destined to go out a port which is shut down are accepted and discarded by that port.

– All of the local VC input and output buffers shall be emptied.

– The receiver shall discard all micropackets except those with TYPE = Reset, Reset_ACK, Initialize, or Initialize_ACK, (i.e., with TYPE = x'2' – x'5').

– The error logging specified in clause 9 shall not occur.

– Administrative actions may clear the error counts accessible by Admin operations (see table 7).

A Link Reset sequence (see 12.1) or Initialize sequence (see 12.2) is the exit from a link shutdown. These may be triggered by a local administrator, or by receipt of a TYPE = Reset or Initialize micropacket.

14 Maintenance and control features

14.1 Timeouts

Table 6 contains a summary of the timeouts, their default value, and the location in this standard discussing the timeout. All of the timeouts shall be programmable, at least to values of 2X, 1/2X, and 1/4X. The mechanisms and procedures used to set values, different from the default values, are outside the scope of this standard.

14.2 Logged events

Table 7 contains a summary of the events that shall be logged, the minimum number of bits for the parameter, and the location in this standard discussing the event. A counter shall not roll over if its maximum value is reached.

Table 6 – Summary of timeouts

Name	Default value	Reference
ACK indication timeout	12 µs	8.2
Activity_Monitor timeout	1 ms	13.1
Credit timeout	2 s	8.1
Hold-off timer	10 s	12.3
Dead-man timer	100 ms	11.2
Stall timeout	2 ms	9.2.4

Table 7 – Summary of logged events

Name	Minimum Number of bits	Reference
ECRC_Error	8	9.1.3
ECRC_Source_Error	8	6.6.3
LCRC_Error	8	9.1.1
Reset_Initialize_Error	1	11.2
Retransmissions	8	8.4
RSEQ_Missing_Error	8	8.2
RSEQ_Out_Of_Range_Error	8	8.2
Skew_Retraining_Error	1	11.2
TSEQ_Error	8	9.1.2
Undefined_TYPE_Value	4	9.1.4
VC[1-2]_Admin_Tail_Error	1/2	9.2.1
VC[0-3]_Credit_Overflow_Error	1/4	8.1
VC[0-3]_Credit_Timeout_Error	1/4	8.1
VC[0-3]_Missing_End_of_Message_Error	1/4	9.2.3
VC[0-3]_Missing_Start_of_Message_Error	1/4	9.2.2
VC[0-3]_RX_VC_Buffer_Overflow	1/4	13.2
VC[0-3]_Stall_Timeout_Error	1/4	9.2.4
VC[0-3]_Undefined_TYPE_Error	1/4	9.1.4
NOTE – The 1/4, and 1/2, entries under the Number of bits column mean that there is one bit for an error, e.g., for VC0_Missing_End_of_Message_Error, and a total of four, or two, errors possible (i.e., one for each VC).		

15 Local electrical interface (optional)

The local electrical interface is an 8-bit interface intended to connect to on-board optical drivers and receivers. Figure 20 shows the components used in a signal path, and table 8 lists the component values. Specifications shall be met when operating with the specified components. The alternative to the local electrical interface is the copper cable interface (see clause 16).

Table 8 – Local interface components

Component	Value	Units	Tolerance
C_a	100	pF	± 5%
C_b	100	pF	± 5%
R_a	55	Ω	± 5%
R_b	55	Ω	± 5%
R_c	55	Ω	± 5%
R_d	75	Ω	± 5%

Differential drivers shall be used on all signal lines, except for the "light-present" signal (see 15.3). Signals in the 'true' or '1' state shall have the xx_Out_p pins more positive than the xx_Out_n pins with a peak-to-peak value within the voltage range specified in table 10 for driver output voltage. The corresponding optical signal shall be 'true' or '1' = 'light-on'. Implementation of some differential Source drivers may require DC termination for correct operation. In these cases a 55 Ω termination (shown as R_a in figure 20) is recommended at the vendor recommended termination voltage. Rise and fall times shall be measured at the 20% and 80% points of the peak-to-peak signal transition. All parameters shall be measured at the Source driver output pins.

The timing for the Source signals shall be as specified in table 9, and shall be measured at the Source driver output pins. During a training sequence, the signals shall be as shown in figure 18.

15.1 Local electrical interface - output

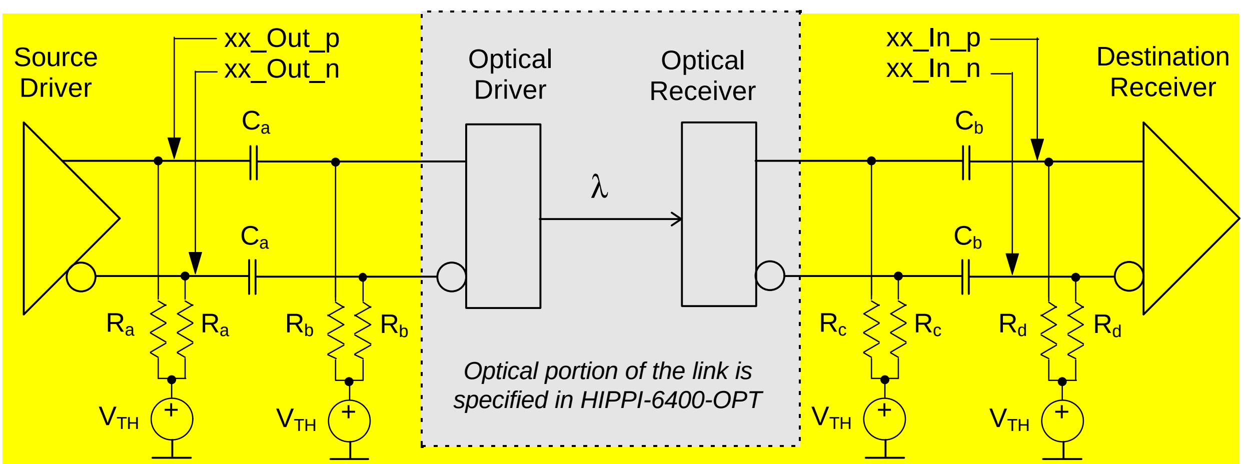


Figure 20 – Local electrical interface with optical components

15.2 Local electrical interface - input

A received 'light-on' optical signal shall indicate 'true' or '1'. The corresponding 'true' or '1' local electrical differential signal shall have the xx_Out_p pins more positive than the xx_Out_n pins. Implementation of some optical receivers may require DC termination for correct operation. In these cases a 55 Ω termination (shown as R_c in figure 20) may be used at the vendor recommended termination voltage. The R_d terminating resistor may be integrated into the Destination receiver, or an external resistor terminated at the vendor recommended termination voltage as shown in figure 20. Rise and fall times shall be measured at the 20% and 80% points of the peak-to-peak signal transition. All parameters are measured at the Destination receiver input pins. The received signals shall be strobed on both edges of the CLOCK signal.

15.3 Light present signal

The activity monitor (see 13.1) shall be driven by a 'light-present' signal from the Destination's optical receiver. A no-light condition shall be indicated by no current into the activity monitor input, allowing it to float to a value of +TBD - +TBD V, Pulling the activity monitor input to a value of +TBD - +TBD V, with a maximum current of TBD ma, shall indicate light present.

Within 1 µs of the activity monitor input going high, the local output signals (with the exception of the CLOCK signal) shall be driven to '0' or 'false', resulting in no light being transmitted on these signals. The local output signals shall remain in this state until the activity monitor input is pulled low. The CLOCK signal shall be continually driven (as specified in 15.1) independent of the activity monitor input level.

NOTES

1 – The activity monitor is available for an open fiber control function, shutting off the light for eye-safety reasons unless a complete (non-broken) optical link is detected. The CLOCK signal, uninterrupted and with a 50% duty cycle, is intended as the pilot signal. Detection of the CLOCK signal would be an indication that the optical path is complete, and hence the other signals can now be driven.

2 – If an open fiber control function is not required, (e.g., the optical power is within eye-safety limits), then the activity monitor input should be grounded.

Table 9 – Local electrical signal timing at Source driver output

Parameter	Value	Units	Comments
CLOCK and CLOCK_2 signals			
CLOCK Period	2	ns	
Tolerance	± 0.4	ps	± 0.02% or 200 ppm
CLOCK Duty Cycle	50	%	
Tolerance	± 50	ps	± 5%
DATA and Control signals			
Bit Period	2	ns	± 0.02% or 200 ppm
Tolerance	± 0.4	ps	± 0.02% or 200 ppm
Duty Cycle	50	%	1010 pattern
Tolerance	± 50	ps	± 5%
FRAME signal			
Period	40	ns	
Tolerance	± 0.8	ps	± 0.002% or 20 ppm
Duty Cycle	50	%	
Tolerance	± 50	ps	± 0.5%

Table 10 – Local electrical interface, Source driver output

Parameter	Max.	Typical	Min.	Units	Comments
V_o	1500		1000	mVp-p	Driver output voltage swing
T_R	320	160	80	ps	Rise time into test load
T_F	320	160	80	ps	Fall time into test load
F_{in}			500	MHz	Operating frequency
Imbalance	40			ps	Driver imbalance skew
T_{JITTER}	122			ps	Total deterministic and random p-p jitter
Source driver timing					
T_{JITTER}	127			ps	Total source p-p jitter
T_{PWD}	100			ps	Total source pulse width distortion
Channel skew	500			ps	Total pair-to-pair skew
NOTE – All measurements are single-ended rather than differential.					

Table 11 – Local electrical interface, Destination receiver input

Parameter	Max.	Typical	Min.	Units	Comments
Input signal parameters					
V_{in}	2700	-	500	mVp-p	Input voltage swing
T_R	480			ps	Rise time at receiver input
T_F	480			ps	Fall time at receiver input
Imbalance	310			ps	Within a signal pair
T_{PWD}	128			ps	Total duty cycle distortion
T_{JITTER}	312			ps	Total deterministic and random p-p jitter
Absolute maximum input voltage					
V_{in}	3400		-700	mV	Input voltage limits
Receiver parameters					
V_{in}	2700	-	150	mVp-p	Input voltage
Imbalance	350			ps	Imbalance skew tolerance (2)
C_{in}	TBD	TBD	TBD	pF	Input capacitance
F_{in}			500	MHz	Input operating frequency
R_{in}		-	75	Ω	Input impedance - single (1)
R_{in}		-	150	Ω	Input impedance - differential (1)
NOTES – 1 Integrated receiver and termination only 2 Additional simulation required to verify operation 3 All measurements are single-ended rather than differential.					

16 Copper cable interface (optional)

The copper cable interface is a 16-bit interface for driving a multi-conductor copper cable for distances up to 50 meters. Figure 21 shows the components used in a signal path and table 12 lists the component values. Specifications shall be met when operating with the specified components and cable. The alternative to the copper cable interface is the local electrical interface (see clause 15).

Table 12 – Copper cable interface components

Component	Value	Units	Tolerance
C_a	100	pF	± 5%
C_b	4	pF	± 5%
R_a	675	Ω	± 5%
R_b	75	Ω	± 5%

16.1 Copper cable interface - output

Differential drivers shall be used on all signal lines. Signals in the 'true' or '1' state shall have the xx_Out_p pin more positive than the xx_Out_n pin with a peak-to-peak value within the voltage range specified in table 14. Rise and fall times shall be measured at the 20% and 80% points of the peak-to-peak signal transition. All parameters shall be measured at the Source driver output pins (see figure 21).

The Source coupling network (i.e., C_a, C_b, and R_a in figure 21) shall implement an equalization network matched to the cable parameters. The equalization network specified is optimized for a 50 meter 150 Ω twin-ax cable, and usable with cables as short as TBD m. Table 12 summarizes the component values, and annex A.6 describes the performance, of the equalization network.

Open issue – What is the shortest length cable we can use with the on-board coupling network?

Open issue - At the March meeting we re-opened the issue of putting the equalizing network in the cable backshell. This would allow customizing the equalizing network for the particular cable, and cable length. The goal is to resolve this issue by April.

The timing for the Source signals shall be as specified in table 13, and shall be measured at the Source driver output pins. During a training sequence, the signals shall be as shown in figure 17.

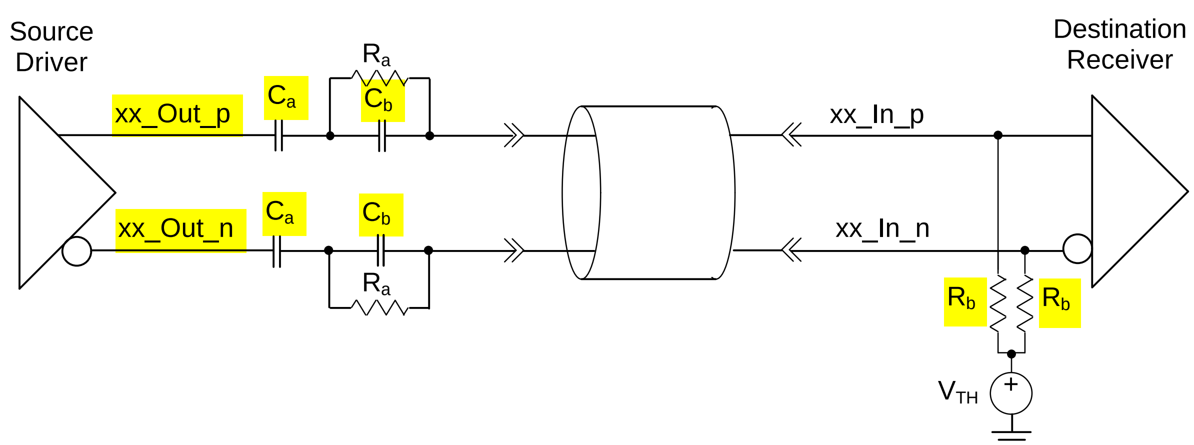


Figure 21 – Copper cable interface signal path

16.2 Copper cable interface - input

Differential receivers shall be used on all signal lines. A received differential signal with the xx_Out_p pin more positive than the xx_Out_n pin shall indicate a 'true' or '1'. Rise and fall times shall be measured at the 20% and 80% points of the peak-to-peak signal transition. All parameters are measured at the Destination receiver input pins. The received signals shall be strobed on both edges of the CLOCK signal. Receivers shall operate correctly when receiving signals meeting the specifications in table 15.

The Destination termination network (i.e., R_b in figure 21) shall terminate the cable with the cable's characteristic impedance, i.e., $R_b = 75\ \Omega \pm 5\%$ (single ended) or $150\ \Omega \pm 5\%$ (differential) to the vendor specific Thevenin voltage (e.g., common mode voltage). The termination network may be integrated into the receiver, or an external network as shown in figure 21.

16.3 CLOCK_2

The phase relationship between the CLOCK_2 and CLOCK signals shall be any constant value. The intended uses of the separate CLOCK and CLOCK_2 signals are:

– to support different skew compensation implementations (e.g., some implementations may prefer to use CLOCK_2, instead of CLOCK, to strobe the signals). The clock signal used to strobe the signals during retraining shall also be used to strobe the signals during regular operation.

– to provide a separate signal that can be monitored for activity (see 13.1) without affecting the signal used for strobing the other signals. If inactivity is detected, the other signals should be ignored to avoid spurious error indications, and an implementation may choose to power down its outputs.

– to provide a free-running clock for systems using phase-locked loops (PLLs) or other implementations that cannot tolerate dropouts of the clock signal.

Table 13 – Copper cable interface signal timing at Source driver output

Parameter	Value	Units	Comments
CLOCK and CLOCK_2 signals			
CLOCK Period	4	ns	
Tolerance	± 0.8	ps	± 0.02% or 200 ppm
CLOCK Duty Cycle	50	%	
Tolerance	± 100	ps	± 5%
DATA and Control signals			
Bit Period	4	ns	± 0.02% or 200 ppm
Tolerance	± 0.8	ps	± 0.02% or 200 ppm
Duty Cycle	50	%	1010 pattern
Tolerance	± 100	ps	± 5%
FRAME signal			
Period	40	ns	
Tolerance	± 0.8	ps	± 0.002% or 20 ppm
Duty Cycle	50	%	
Tolerance	± 100	ps	± 0.5%

Table 14 – Copper cable interface, Source driver output

Parameter	Max.	Typical	Min.	Units	Comments
V_o	2700	2500	2200	mVp-p	Driver output voltage
T_R	320	160	80	ps	Rise time into test load
T_F	320	160	80	ps	Fall time into test load
F_{in}			250	MHz	Operating frequency
Imbalance	40			ps	Driver imbalance skew
T_{JITTER}	122			ps	Total deterministic and random p-p jitter
R_o	TBD	TBD	TBD	Ω	Output impedance
Source driver timing					
T_{JITTER}	127			ps	Total source p-p jitter
T_{PWD}	100			ps	Total source pulse width distortion
Channel skew	500			ps	Total pair-to-pair skew
NOTE – All measurements are single-ended rather than differential.					

Table 15 – Copper cable interface, Destination receiver input

Parameter	Max.	Typical	Min.	Units	Comments
Input signal parameters					
V_{IN}			200	mVp-p	Includes 50 mV noise margin
T_R	480			ps	Rise time at the receiver input
T_F	480			ps	Fall time at the receiver input
Imbalance	310			ps	Within pair skew, includes 40 ps margin
T_{PWD}	128			ps	Total duty cycle distortion
T_{JITTER}	312			ps	Total peak-to-peak, includes 38 ps margin
Absolute maximum input voltages					
V_{in}	3400		-700	mV	Input voltages
Receiver parameters					
V_{in}	2700	-	150	mVp-p	Input voltage
Imbalance	350			ps	Imbalance skew tolerance (2)
C_{in}	TBD	TBD	TBD	pF	Input capacitance
F_{in}			250	MHz	Input operating frequency
R_{in}		-	75	Ω	Input impedance - single (1)
R_{in}		-	150	Ω	Input impedance - differential (1)
NOTES – 1 Integrated receiver and termination only 2 Additional simulation required to verify operation 3 All measurements are single-ended rather than differential.					

16.4 Copper cable connectors

The receptacle shall be Berg Micropax 100 position, part number 72546-40x or equivalent (the "x" depends upon the board thickness). A right-angle mount receptacle is shown in figure 24; other mounting methods may be used. The mating cable connector, as shown in figure 25 shall be a Berg Micropax 100 position, part number 72524-001, or equivalent. Figure 25 shows a cable connector with a straight exit; other exit configurations may be used.

NOTE – Berg Electronics connectors are examples of suitable products available commercially. This information is given for the convenience of users of this standard and does not constitute an endorsement by ANSI, or other publisher of this standard, of these products.

The receptacle pin assignments shall be as shown in figure 23; pins labeled n.c. shall not be connected. The mating cable connectors shall be wired as shown in table 17.

These connector specifications shall apply for a minimum of 1000 mating cycles.

Each pin shall have a ≥ 1 A current capability, with the total current capability for all pins simultaneously shall be ≥ 5 A.

The connectors shall provide RFI/EMI shielding sufficient to pass all appropriate compliance tests. When mated, the receptacle housing shall provide the ground path for the connector backshell.

Signal attenuation shall be ≤ 0.1 dB. When multiple pairs are driven differentially with a 100 ps risetime (20% – 80%) pulse, near end crosstalk shall be ≤ 12%.

Connector thickness shall be ≤ 0.75".

Jackscrews with 4-40 threads shall be used to hold the connectors in the mated position.

16.5 Copper cable specifications

The cable assembly (i.e., cable and connectors) shall provide differential paths for 46 signals, 23 in each direction. Cable assembly length is determined by cable quality and environmental factors. All cable assemblies shall meet the specifications in table 16.

The cable shall have an outside diameter ≤ 0.665 in (16.9 mm) and a bend radius ≤ 6 in (152 mm).

The cable shall provide individual shields, or equivalent, for each differential path. These individual shields shall be floating, i.e., isolated from each other, from the overall shield, and from the connector.

There shall be an overall shield. As shown in figure 22, at one end of the cable the overall shield shall be connected to pins 51 and 100 through a total capacitance of 0.4 µf at 50 V. At the other end of the cable the overall shield shall be directly connected to pins 51 and 100. The overall shield shall be insulated from the connector backshell at both ends.

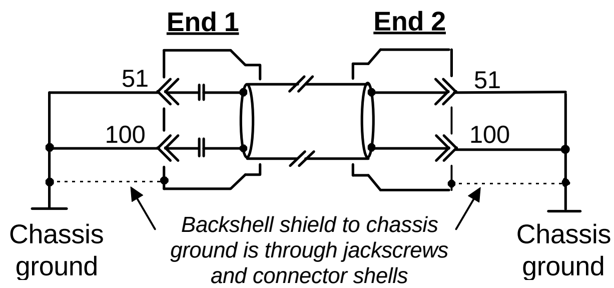


Figure 22 – Connecting the overall shield

Table 16 – Copper cable assembly electrical specifications

Parameter	Max.	Typ.	Min.	Units	Comments
Z_0	165	150	135	Ω	Differential impedance (tolerance ± 10%)
$Loss_{AC}$	-20			dB	AC loss @ 500 MHz
$Loss_{DC}$	2			Ω	DC resistance
V_{XTALK}	100			mV•ns	Reverse cross talk voltage
V_o			200	mVp-p	Single ended peak-to-peak output voltage
V_{EYE}			400	mVp-p	Eye pattern peak-to-peak voltage opening
T_{JITTER}	180			ps	Deterministic jitter peak-to-peak
Channel Skew	5000			ps	Channel-to-channel skew
Imbalance Skew	250			ps	Imbalance skew within a signal pair
NOTE – All measurements are single-ended rather than differential.					

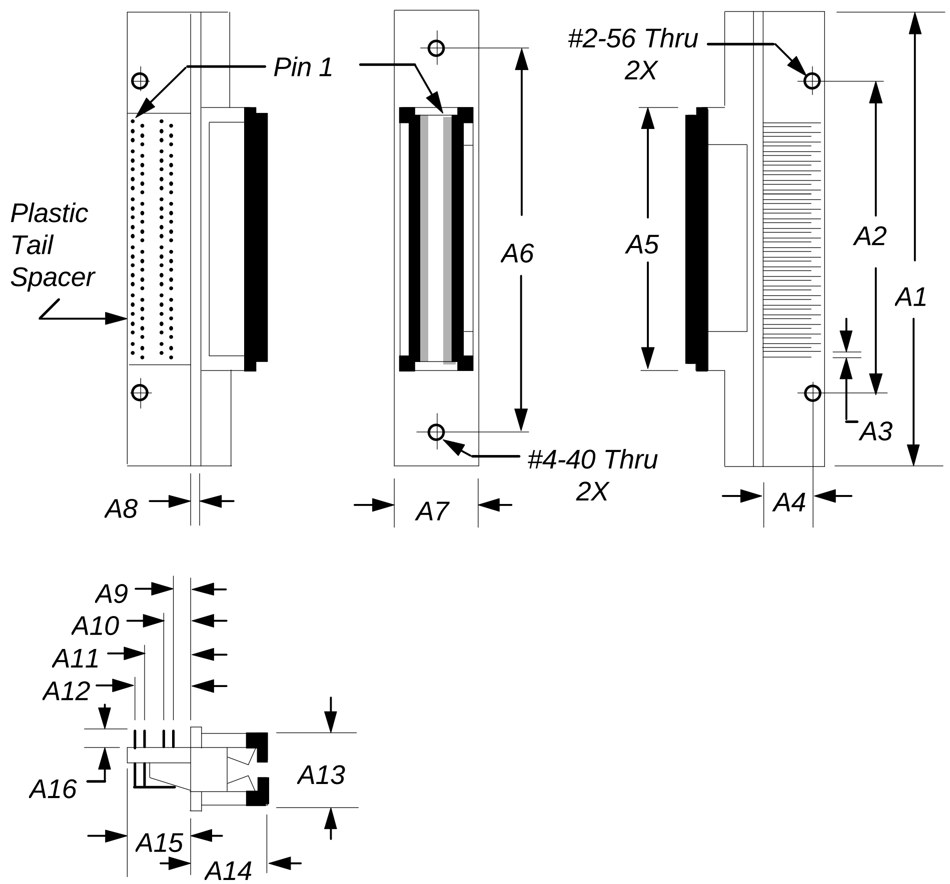
Chassis Ground	51	1	CLOCK_2_Out_p
D00_Out_p	52	2	CLOCK_2_Out_n
D00_Out_n	53	3	D08_Out_p
D01_Out_p	54	4	D08_Out_n
D01_Out_n	55	5	D09_Out_p
D02_Out_p	56	6	D09_Out_n
D02_Out_n	57	7	D10_Out_p
D03_Out_p	58	8	D10_Out_n
D03_Out_n	59	9	D11_Out_p
D04_Out_p	60	10	D11_Out_n
D04_Out_n	61	11	D12_Out_p
D05_Out_p	62	12	D12_Out_n
D05_Out_n	63	13	D13_Out_p
D06_Out_p	64	14	D13_Out_n
D06_Out_n	65	15	D14_Out_p
D07_Out_p	66	16	D14_Out_n
D07_Out_n	67	17	D15_Out_p
C0_Out_p	68	18	D15_Out_n
C0_Out_n	69	19	C2_Out_p
C1_Out_p	70	20	C2_Out_n
C1_Out_n	71	21	C3_Out_p
CLOCK_Out_p	72	22	C3_Out_n
CLOCK_Out_n	73	23	FRAME_Out_p
n.c.	74	24	FRAME_Out_n
n.c.	75	25	n.c.
n.c.	76	26	n.c.
n.c.	77	27	FRAME_In_n
CLOCK_In_n	78	28	FRAME_In_p
CLOCK_In_p	79	29	C3_In_n
C1_In_n	80	30	C3_In_p
C1_In_p	81	31	C2_In_n
C0_In_n	82	32	C2_In_p
C0_In_p	83	33	D15_In_n
D07_In_n	84	34	D15_In_p
D07_In_p	85	35	D14_In_n
D06_In_n	86	36	D14_In_p
D06_In_p	87	37	D13_In_n
D05_In_n	88	38	D13_In_p
D05_In_p	89	39	D12_In_n
D04_In_n	90	40	D12_In_p
D04_In_p	91	41	D11_In_n
D03_In_n	92	42	D11_In_p
D03_In_p	93	43	D10_In_n
D02_In_n	94	44	D10_In_p
D02_In_p	95	45	D09_In_n
D01_In_n	96	46	D09_In_p
D01_In_p	97	47	D08_In_n
D00_In_n	98	48	D08_In_p
D00_In_p	99	49	CLOCK_2_In_n
Chassis Ground	100	50	CLOCK_2_In_p

NOTE – n.c. = no connection allowed

Figure 23 – Receptacle pin assignments

Table 17 – Cable layout

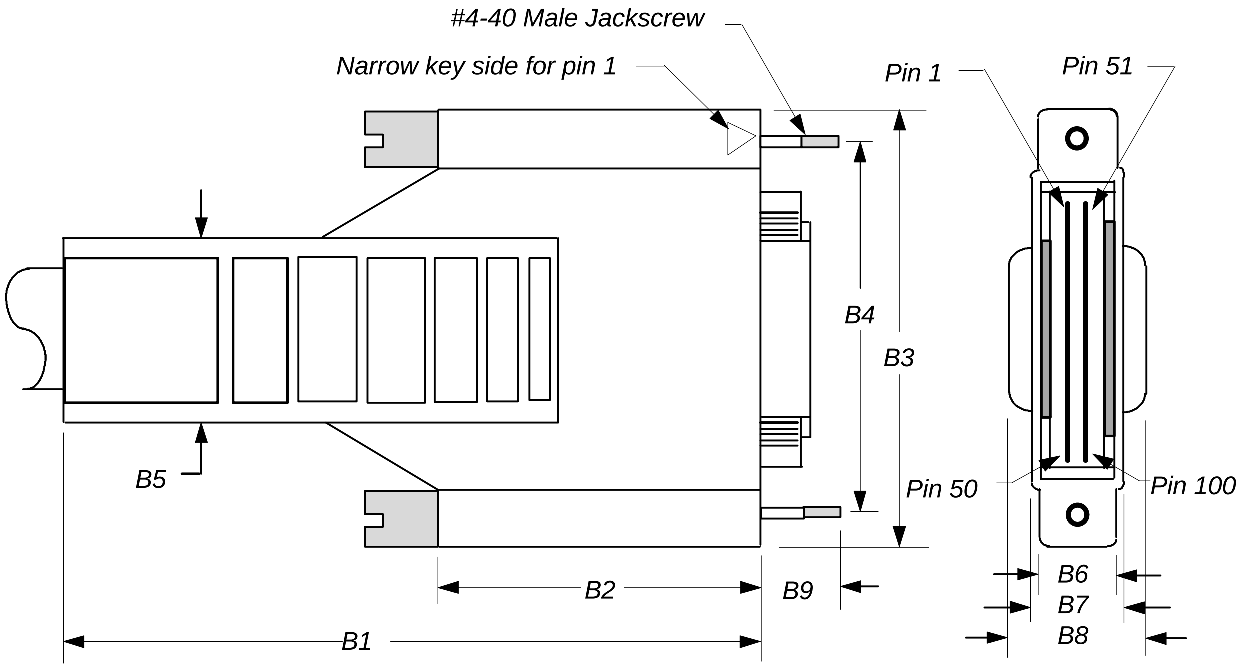
End 1			**End 2**	
Pin p	**Pin n**	**Signal name**	**Pin p**	**Pin n**
52	53	D00 —>	99	98
54	55	D01 —>	97	96
56	57	D02 —>	95	94
58	59	D03 —>	93	92
60	61	D04 —>	91	90
62	63	D05 —>	89	88
64	65	D06 —>	87	86
66	67	D07 —>	85	84
3	4	D08 —>	48	47
5	6	D09 —>	46	45
7	8	D10 —>	44	43
9	10	D11 —>	42	41
11	12	D12 —>	40	39
13	14	D13 —>	38	37
15	16	D14 —>	36	35
17	18	D15 —>	34	33
68	69	C0 —>	83	82
70	71	C1 —>	81	80
19	20	C2 —>	32	31
21	22	C3 —>	30	29
23	24	FRAME —>	28	27
72	73	CLOCK —>	79	78
1	2	CLOCK_2 —>	50	49
99	98	<— D00	52	53
97	96	<— D01	54	55
95	94	<— D02	56	57
93	92	<— D03	58	59
91	90	<— D04	60	61
89	88	<— D05	62	63
87	86	<— D06	64	65
85	84	<— D07	66	67
48	47	<— D08	3	4
46	45	<— D09	5	6
44	43	<— D10	7	8
42	41	<— D11	9	10
40	39	<— D12	11	12
38	37	<— D13	13	14
36	35	<— D14	15	16
34	33	<— D15	17	18
83	82	<— C0	68	69
81	80	<— C1	70	71
32	31	<— C2	19	20
30	29	<— C3	21	22
28	27	<— FRAME	23	24
79	78	<— CLOCK	72	73
50	49	<— CLOCK_2	1	2
51 ac	100 ac	Overall shield	51	100



Dimension	mm	inches
A1	60.20	2.370
A2	41.27	1.625
A3	0.63 Typical	0.025 Typical
A4	6.350	0.250
A5	34.93	1.375
A6	50.80	2.000
A7	11.18	0.440
A8	1.270	0.050

Dimension	mm	inches
A9	2.540	0.100
A10	3.810	0.150
A11	6.350	0.250
A12	7.620	0.300
A13	9.520	0.375
A14	10.03	0.395
A15	8.130	0.320
A16	*Dependent on board thickness*	

Figure 24 – Receptacle



Dimension	mm	inches
B1	96.28 Max	3.80 Max
B2	43.18	1.70
B3	58.67 Max	2.31 Max
B4	50.80	2.00
B5	25.40	1.00
B6	10.92	0.43
B7	12.70	0.50
B8	19.05 Max	0.750 Max
B9	10.77	0.42

Figure 25 – Cable connector

Annex A
(informative)

Implementation comments

A.1 4b/5b encoding and decoding

Encoding the 4-bit code groups into 5-bit transmission codes may be implemented as shown in the left portion of the example in figure A.1. Decoding the 4-bit code from the 5-bit code may be implemented as shown in the right portion of figure A.1. The specification for the encoding and decoding is in 10.2 and 10.3.

A.2 Frequency differences between Source and Destination

Although the two ends of a HIPPI-6400 link run at nominally the same speed, there can be very slight differences in clock frequency due to inaccuracy of the crystal oscillators at each end. If a transmitter is allowed to send an very long burst of continuous traffic, this will eventually cause a receiver to overrun if that receiver's clock is slightly slower than the transmitter's clock.

To prevent this condition, the length of continuous data transmission is limited by inserting non-data micropackets (training sequences in HIPPI-6400-PH) periodically. The frequency of training sequences is determined by the potential inaccuracy of the oscillators and the amount of drift the receiver can tolerate. With ± 200 ppm of frequency error (see 15.1), the total clock error could be as large as 400 ppm, since the sender and receiver could be off in opposite directions. Allowing a drift of 4 ns before correction, takes 4 ns x (1/400 ppm) = 10 µs. Hence, the requirement that HIPPI-6400-PH transmitters insert retraining sequences at least every 10 µs.

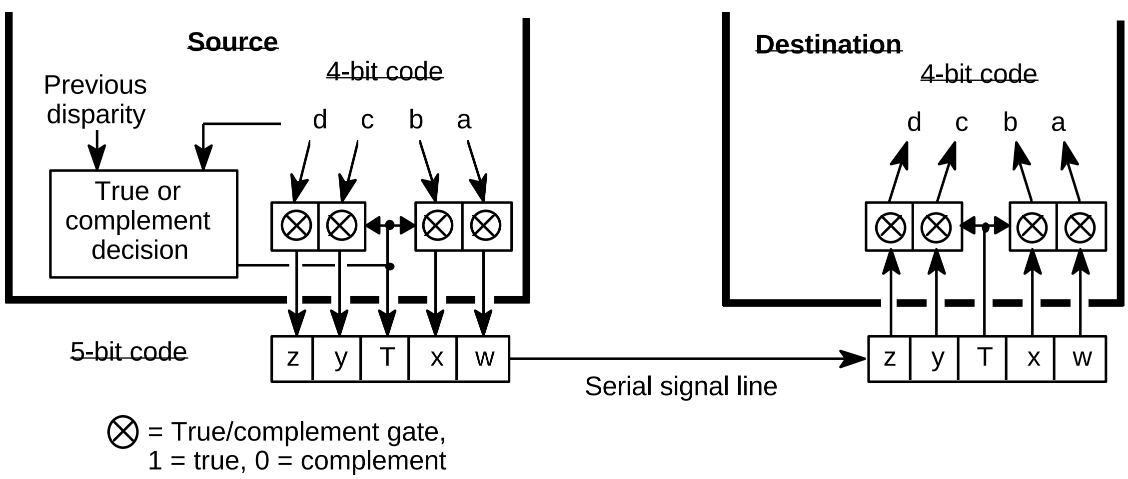


Figure A.1 – Encode / decode circuit example

A.3 LCRC parallel implementation

The LCRC specified in 6.6.2 and figure 11 is based on a bit-by-bit serial implementation. Parallel implementations may be used as long as they produce the same results as the serial example. Tables A.3 and A.4 give equations for 16-bit and 64-bit parallel LCRC implementations, useful for LCRC generation as shown in figure A.2. Table A.5 gives 80-bit parallel equations for LCRC checking, as shown in figure A.3. Other parallel widths may be used, these are just examples.

For these LCRC equations, c63 through c48 are the flip-flops shown in figure 11, and the resultant LCRC control bits. b*n* are the bits which must be delivered to the parallel equation simultaneously. b0 is the first bit which would have been supplied to the serial implementation. The r*n* bits are the all 1's seed value, and the intermediate results from the Partial LCRC Register.

A.3.1 Parallel LCRC generator

Parallel LCRC generation can be accomplished by cascading 16-bit parallel equations and 64-bit parallel equations as shown in Figure A.2. Four clock periods are used to produce the LCRC value for a micropacket. Table A.1 summarizes the input bits for each clock period.

Table A.1 – Parallel LCRC input bits

Clock Period	b79:64	b63:00	Mux output
1	c00–c15	d00.0–d07.7	x'FFFF'
2	c16–c31	d08.0–d15.7	partial
3	c32–c47	d16.0–d23.7	partial
4	c48–c63	d24.0–d31.7	partial

a During the first period, the c00–c15 are applied to the 16-bit equations, and the 64 bits d00.0–d07.7 are applied to the 64-bit LCRC equations. Note that for this first cycle only, the multiplexer is set to force x'FFFF' as the 16-bit partial LCRC value, (i.e., initializing with a value of all ones). The register is clocked after the signals have settled.

b During the second period, c16–c31 are applied to the 16-bit equations, and d08.0–d15.7 are applied to 64-bit equations. The register is clocked a second time.

c During the third period, c32–c47 are applied to the 16-bit equations, and d16.0–d23.7 are applied to the 64-bit equations. The register is clocked a third time.

d During the fourth, and final, period, the c48–c63 values presented to the 16-bit equations are immaterial (they are just included for consistency with the LCRC checker), and d24.0–d31.7 are applied to the 64-bit equations – the register is not clocked. After appropriate settling time, the LCRC is available as c63–c48 (c63 is the msb).

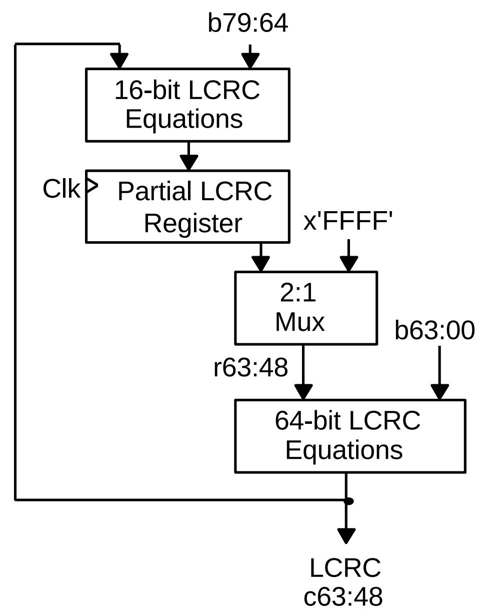


Figure A.2 – Parallel LCRC generator example

A.3.2 Parallel LCRC checker

Like the LCRC generator, the LCRC checker uses four clock periods to produce the LCRC check value. The LCRC checker can use a single set of 80-bit equations as shown in figure A.3 rather than cascading 16-bit and 64-bit equations. The difference between the generator and checker is that the generator does not include the LCRC bits (c63:48) in the calculation's final step, while the checker includes them. A final LCRC value of x'0000' means no error; x'06A9' means a stomp code. The input bits are also summarized in table A.1, and the time steps are essentially the same.

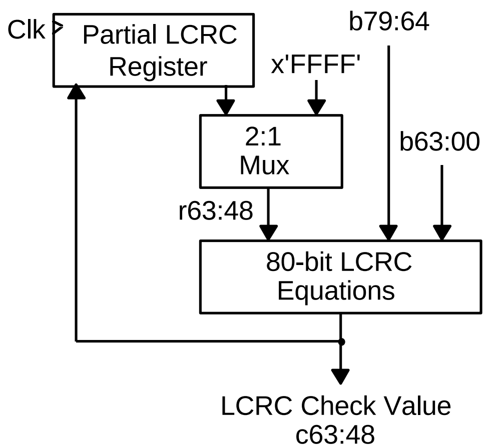


Figure A.3 – Parallel LCRC checker example

A.4 ECRC parallel implementation

The ECRC specified in 6.6.3 and figure 12 is based on a bit-by-bit serial implementation. Parallel implementations may be used as long as they produce the same results as the serial example. Table A.6 gives equations for a 64-bit parallel ECRC implementation as shown in figure A.4. Other parallel widths may be used, this is just an example.

For these ECRC equations, c47 through c32 are the flip-flops shown in figure 12, and the resultant ECRC control bits. b*n* are the bits which must be delivered to the parallel equation simultaneously. b0 is the first bit which would have been supplied to the serial implementation. The r*n* bits are the all 1's seed value, and the intermediate results from the Partial ECRC Register. Four partial ECRC registers are required since the ECRC is continued across multiple micropackets, and the micropackets from different VC's can be interleaved. Four clock periods are used to produce the ECRC value for a micropacket. Table A.2 summarizes the input bits for each clock period.

Table A.2 – Parallel ECRC input bits

Clock Period	b63:00	Mux output
1	d00.0–d07.7	*(see text)*
2	d08.0–d15.7	partial
3	d16.0–d23.7	partial
4	d24.0–d31.7	partial

a During the first period, the 64 bits d00.0–d07.7 are applied to the 64-bit ECRC equations. Note that for this first cycle only, and only if this is the first micropacket of a Message, the multiplexer is set to force x'FFFF' as the 16-bit partial ECRC value, (i.e., initializing with a value of all ones). The appropriate VC partial register is clocked after the signals have settled.

b During the second period, d08.0–d15.7 are applied to 64-bit equations. The appropriate register is clocked a second time.

c During the third period, d16.0–d23.7 are applied to the 64-bit equations. The appropriate register is clocked a third time.

d During the fourth, and final, period, d24.0–d31.7 are applied to the 64-bit equations. After appropriate settling time, and without clocking the register, the ECRC is available as c63–c48 (c63 is the msb). Then, after capturing this ECRC, the appropriate register is again clocked to accumulate all data for the entire message.

A.5 Undetected errors

Simulations have shown that all cases of up to five simultaneous bit errors in a micropacket are detected. Four cases of 4-bit errors are not detected by LCRC or ECRC errors, but are detected by other tests, e.g., bad TSEQ values.

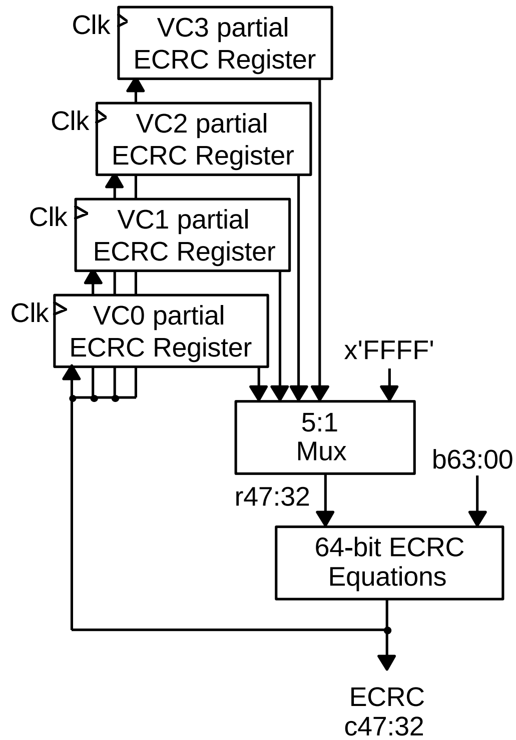


Figure A.4 – Parallel ECRC example

Table A.3 – 16-bit LCRC generator equations

Output	Exclusive OR these bits together															
r63	b79	b75	b71	b68	b67	c63	c59	c55	c52	c51						
r62	b78	b74	b70	b67	b66	c62	c58	c54	c51	c50						
r61	b77	b73	b69	b66	b65	c61	c57	c53	c50	c49						
r60	b76	b72	b68	b65	b64	c60	c56	c52	c49	c48						
r59	b75	b71	b67	b64	c59	c55	c51	c48								
r58	b79	b75	b74	b71	b70	b68	b67	b66	c63	c59	c58	c55	c54	c52	c51	c50
r57	b78	b74	b73	b70	b69	b67	b66	b65	c62	c58	c57	c54	c53	c51	c50	c49
r56	b77	b73	b72	b69	b68	b66	b65	b64	c61	c57	c56	c53	c52	c50	c49	c48
r55	b76	b72	b71	b68	b67	b65	b64	c60	c56	c55	c52	c51	c49	c48		
r54	b75	b71	b70	b67	b66	b64	c59	c55	c54	c51	c50	c48				
r53	b74	b70	b69	b66	b65	c58	c54	c53	c50	c49						
r52	b73	b69	b68	b65	b64	c57	c53	c52	c49	c48						
r51	b79	b75	b72	b71	b64	c63	c59	c56	c55	c48						
r50	b78	b74	b71	b70	c62	c58	c55	c54								
r49	b77	b73	b70	b69	c61	c57	c54	c53								
r48	b76	b72	b69	b68	c60	c56	c53	c52								

Table A.4 – 64-bit LCRC generator equations

Output	Exclusive OR these bits together
c63	b63 b59 b55 b52 b51 b44 b43 b41 b37 b36 b35 b31 b30 b28 b21 b15 b14 b12 b11 b8 b7 b5 b0 r63 r62 r60 r59 r56 r55 r53 r48
c62	b62 b58 b54 b51 b50 b43 b42 b40 b36 b35 b34 b30 b29 b27 b20 b14 b13 b11 b10 b7 b6 b4 r62 r61 r59 r58 r55 r54 r52
c61	b61 b57 b53 b50 b49 b42 b41 b39 b35 b34 b33 b29 b28 b26 b19 b13 b12 b10 b9 b6 b5 b3 r61 r60 r58 r57 r54 r53 r51
c60	b60 b56 b52 b49 b48 b41 b40 b38 b34 b33 b32 b28 b27 b25 b18 b12 b11 b9 b8 b5 b4 b2 r60 r59 r57 r56 r53 r52 r50
c59	b59 b55 b51 b48 b47 b40 b39 b37 b33 b32 b31 b27 b26 b24 b17 b11 b10 b8 b7 b4 b3 b1 r59 r58 r56 r55 r52 r51 r49
c58	b63 b59 b58 b55 b54 b52 b51 b50 b47 b46 b44 b43 b41 b39 b38 b37 b35 b32 b28 b26 b25 b23 b21 b16 b15 b14 b12 b11 b10 b9 b8 b6 b5 b3 b2 r63 r62 r60 r59 r58 r57 r56 r54 r53 r51 r50
c57	b62 b58 b57 b54 b53 b51 b50 b49 b46 b45 b43 b42 b40 b38 b37 b36 b34 b31 b27 b25 b24 b22 b20 b15 b14 b13 b11 b10 b9 b8 b7 b5 b4 b2 b1 r63 r62 r61 r59 r58 r57 r56 r55 r53 r52 r50 r49
c56	b61 b57 b56 b53 b52 b50 b49 b48 b45 b44 b42 b41 b39 b37 b36 b35 b33 b30 b26 b24 b23 b21 b19 b14 b13 b12 b10 b9 b8 b7 b6 b4 b3 b1 b0 r62 r61 r60 r58 r57 r56 r55 r54 r52 r51 r49 r48
c55	b60 b56 b55 b52 b51 b49 b48 b47 b44 b43 b41 b40 b38 b36 b35 b34 b32 b29 b25 b23 b22 b20 b18 b13 b12 b11 b9 b8 b7 b6 b5 b3 b2 b0 r61 r60 r59 r57 r56 r55 r54 r53 r51 r50 r48
c54	b59 b55 b54 b51 b50 b48 b47 b46 b43 b42 b40 b39 b37 b35 b34 b33 b31 b28 b24 b22 b21 b19 b17 b12 b11 b10 b8 b7 b6 b5 b4 b2 b1 r60 r59 r58 r56 r55 r54 r53 r52 r50 r49
c53	b58 b54 b53 b50 b49 b47 b46 b45 b42 b41 b39 b38 b36 b34 b33 b32 b30 b27 b23 b21 b20 b18 b16 b11 b10 b9 b7 b6 b5 b4 b3 b1 b0 r59 r58 r57 r55 r54 r53 r52 r51 r49 r48
c52	b57 b53 b52 b49 b48 b46 b45 b44 b41 b40 b38 b37 b35 b33 b32 b31 b29 b26 b22 b20 b19 b17 b15 b10 b9 b8 b6 b5 b4 b3 b2 b0 r63 r58 r57 r56 r54 r53 r52 r51 r50 r48
c51	b63 b59 b56 b55 b48 b47 b45 b41 b40 b39 b35 b34 b32 b25 b19 b18 b16 b15 b12 b11 b9 b4 b3 b2 b1 b0 r63 r60 r59 r57 r52 r51 r50 r49 r48
c50	b62 b58 b55 b54 b47 b46 b44 b40 b39 b38 b34 b33 b31 b24 b18 b17 b15 b14 b11 b10 b8 b3 b2 b1 b0 r63 r62 r59 r58 r56 r51 r50 r49 r48
c49	b61 b57 b54 b53 b46 b45 b43 b39 b38 b37 b33 b32 b30 b23 b17 b16 b14 b13 b10 b9 b7 b2 b1 b0 r62 r61 r58 r57 r55 r50 r49 r48
c48	b60 b56 b53 b52 b45 b44 b42 b38 b37 b36 b32 b31 b29 b22 b16 b15 b13 b12 b9 b8 b6 b1 b0 r63 r61 r60 r57 r56 r54 r49 r48

Table A.5 – 80-bit LCRC checker equations

Output	Exclusive OR these bits together
c63	b79 b75 b71 b68 b67 b60 b59 b57 b53 b52 b51 b47 b46 b44 b37 b31 b30 b28 b27 b24 b23 b21 b16 b15 b14 b13 b12 b9 b7 b5 b4 b2 r63 r62 r61 r60 r57 r55 r53 r52 r50
c62	b78 b74 b70 b67 b66 b59 b58 b56 b52 b51 b50 b46 b45 b43 b36 b30 b29 b27 b26 b23 b22 b20 b15 b14 b13 b12 b11 b8 b6 b4 b3 b1 r63 r62 r61 r60 r59 r56 r54 r52 r51 r49
c61	b77 b73 b69 b66 b65 b58 b57 b55 b51 b50 b49 b45 b44 b42 b35 b29 b28 b26 b25 b22 b21 b19 b14 b13 b12 b11 b10 b7 b5 b3 b2 b0 r62 r61 r60 r59 r58 r55 r53 r51 r50 r48
c60	b76 b72 b68 b65 b64 b57 b56 b54 b50 b49 b48 b44 b43 b41 b34 b28 b27 b25 b24 b21 b20 b18 b13 b12 b11 b10 b9 b6 b4 b2 b1 r61 r60 r59 r58 r57 r54 r52 r50 r49
c59	b75 b71 b67 b64 b63 b56 b55 b53 b49 b48 b47 b43 b42 b40 b33 b27 b26 b24 b23 b20 b19 b17 b12 b11 b10 b9 b8 b5 b3 b1 b0 r60 r59 r58 r57 r56 r53 r51 r49 r48
c58	b79 b75 b74 b71 b70 b68 b67 b66 b63 b62 b60 b59 b57 b55 b54 b53 b51 b48 b44 b42 b41 b39 b37 b32 b31 b30 b28 b27 b26 b25 b24 b22 b21 b19 b18 b15 b14 b13 b12 b11 b10 b8 b5 b0 r63 r62 r61 r60 r59 r58 r56 r53 r48
c57	b78 b74 b73 b70 b69 b67 b66 b65 b62 b61 b59 b58 b56 b54 b53 b52 b50 b47 b43 b41 b40 b38 b36 b31 b30 b29 b27 b26 b25 b24 b23 b21 b20 b18 b17 b14 b13 b12 b11 b10 b9 b7 b4 r62 r61 r60 r59 r58 r57 r55 r52
c56	b77 b73 b72 b69 b68 b66 b65 b64 b61 b60 b58 b57 b55 b53 b52 b51 b49 b46 b42 b40 b39 b37 b35 b30 b29 b28 b26 b25 b24 b23 b22 b20 b19 b17 b16 b13 b12 b11 b10 b9 b8 b6 b3 r61 r60 r59 r58 r57 r56 r54 r51
c55	b76 b72 b71 b68 b67 b65 b64 b63 b60 b59 b57 b56 b54 b52 b51 b50 b48 b45 b41 b39 b38 b36 b34 b29 b28 b27 b25 b24 b23 b22 b21 b19 b18 b16 b15 b12 b11 b10 b9 b8 b7 b5 b2 r63 r60 r59 r58 r57 r56 r55 r53 r50
c54	b75 b71 b70 b67 b66 b64 b63 b62 b59 b58 b56 b55 b53 b51 b50 b49 b47 b44 b40 b38 b37 b35 b33 b28 b27 b26 b24 b23 b22 b21 b20 b18 b17 b15 b14 b11 b10 b9 b8 b7 b6 b4 b1 r63 r62 r59 r58 r57 r56 r55 r54 r52 r49
c53	b74 b70 b69 b66 b65 b63 b62 b61 b58 b57 b55 b54 b52 b50 b49 b48 b46 b43 b39 b37 b36 b34 b32 b27 b26 b25 b23 b22 b21 b20 b19 b17 b16 b14 b13 b10 b9 b8 b7 b6 b5 b3 b0 r62 r61 r58 r57 r56 r55 r54 r53 r51 r48
c52	b73 b69 b68 b65 b64 b62 b61 b60 b57 b56 b54 b53 b51 b49 b48 b47 b45 b42 b38 b36 b35 b33 b31 b26 b25 b24 b22 b21 b20 b19 b18 b16 b15 b13 b12 b9 b8 b7 b6 b5 b4 b2 r63 r61 r60 r57 r56 r55 r54 r53 r52 r50
c51	b79 b75 b72 b71 b64 b63 b61 b57 b56 b55 b51 b50 b48 b41 b35 b34 b32 b31 b28 b27 b25 b20 b19 b18 b17 b16 b13 b11 b9 b8 b6 b3 b2 b1 r61 r59 r57 r56 r54 r51 r50 r49
c50	b78 b74 b71 b70 b63 b62 b60 b56 b55 b54 b50 b49 b47 b40 b34 b33 b31 b30 b27 b26 b24 b19 b18 b17 b16 b15 b12 b10 b8 b7 b5 b2 b1 b0 r63 r60 r58 r56 r55 r53 r50 r49 r48
c49	b77 b73 b70 b69 b62 b61 b59 b55 b54 b53 b49 b48 b46 b39 b33 b32 b30 b29 b26 b25 b23 b18 b17 b16 b15 b14 b11 b9 b7 b6 b4 b1 b0 r63 r62 r59 r57 r55 r54 r52 r49 r48
c48	b76 b72 b69 b68 b61 b60 b58 b54 b53 b52 b48 b47 b45 b38 b32 b31 b29 b28 b25 b24 b22 b17 b16 b15 b14 b13 b10 b8 b6 b5 b3 b0 r63 r62 r61 r58 r56 r54 r53 r51 r48

Table A.6 – 64-bit ECRC generator / checker equations

Output	Exclusive OR these bits together
c47	b63 b59 b55 b51 b50 b48 b43 b42 b40 b37 b35 b34 b32 b31 b29 b26 b22 b20 b19 b18 b17 b13 b11 b10 b7 b6 b4 b3 b2 b1 r45 r43 r42 r39 r38 r36 r35 r34 r33
c46	b63 b62 b59 b58 b55 b54 b51 b49 b48 b47 b43 b41 b40 b39 b37 b36 b35 b33 b32 b30 b29 b28 b26 b25 b22 b21 b20 b16 b13 b12 b11 b9 b7 b5 b4 b0 r45 r44 r43 r41 r39 r37 r36 r32
c45	b62 b61 b58 b57 b54 b53 b50 b48 b47 b46 b42 b40 b39 b38 b36 b35 b34 b32 b31 b29 b28 b27 b25 b24 b21 b20 b19 b15 b12 b11 b10 b8 b6 b4 b3 r47 r44 r43 r42 r40 r38 r36 r35
c44	b63 b61 b60 b59 b57 b56 b55 b53 b52 b51 b50 b49 b48 b47 b46 b45 b43 b42 b41 b40 b39 b38 b33 b32 b30 b29 b28 b27 b24 b23 b22 b17 b14 b13 b9 b6 b5 b4 b1 r46 r45 r41 r38 r37 r36 r33
c43	b62 b60 b59 b58 b56 b55 b54 b52 b51 b50 b49 b48 b47 b46 b45 b44 b42 b41 b40 b39 b38 b37 b32 b31 b29 b28 b27 b26 b23 b22 b21 b16 b13 b12 b8 b5 b4 b3 b0 r45 r44 r40 r37 r36 r35 r32
c42	b61 b59 b58 b57 b55 b54 b53 b51 b50 b49 b48 b47 b46 b45 b44 b43 b41 b40 b39 b38 b37 b36 b31 b30 b28 b27 b26 b25 b22 b21 b20 b15 b12 b11 b7 b4 b3 b2 r47 r44 r43 r39 r36 r35 r34
c41	b60 b58 b57 b56 b54 b53 b52 b50 b49 b48 b47 b46 b45 b44 b43 b42 b40 b39 b38 b37 b36 b35 b30 b29 b27 b26 b25 b24 b21 b20 b19 b14 b11 b10 b6 b3 b2 b1 r46 r43 r42 r38 r35 r34 r33
c40	b59 b57 b56 b55 b53 b52 b51 b49 b48 b47 b46 b45 b44 b43 b42 b41 b39 b38 b37 b36 b35 b34 b29 b28 b26 b25 b24 b23 b20 b19 b18 b13 b10 b9 b5 b2 b1 b0 r45 r42 r41 r37 r34 r33 r32
c39	b58 b56 b55 b54 b52 b51 b50 b48 b47 b46 b45 b44 b43 b42 b41 b40 b38 b37 b36 b35 b34 b33 b28 b27 b25 b24 b23 b22 b19 b18 b17 b12 b9 b8 b4 b1 b0 r44 r41 r40 r36 r33 r32
c38	b57 b55 b54 b53 b51 b50 b49 b47 b46 b45 b44 b43 b42 b41 b40 b39 b37 b36 b35 b34 b33 b32 b27 b26 b24 b23 b22 b21 b18 b17 b16 b11 b8 b7 b3 b0 r43 r40 r39 r35 r32
c37	b56 b54 b53 b52 b50 b49 b48 b46 b45 b44 b43 b42 b41 b40 b39 b38 b36 b35 b34 b33 b32 b31 b26 b25 b23 b22 b21 b20 b17 b16 b15 b10 b7 b6 b2 r47 r42 r39 r38 r34
c36	b55 b53 b52 b51 b49 b48 b47 b45 b44 b43 b42 b41 b40 b39 b38 b37 b35 b34 b33 b32 b31 b30 b25 b24 b22 b21 b20 b19 b16 b15 b14 b9 b6 b5 b1 r47 r46 r41 r38 r37 r33
c35	b63 b59 b55 b54 b52 b47 b46 b44 b41 b39 b38 b36 b35 b33 b30 b26 b24 b23 b22 b21 b17 b15 b14 b11 b10 b8 b7 b6 b5 b3 b2 b1 b0 r47 r46 r43 r42 r40 r39 r38 r37 r35 r34 r33 r32
c34	b62 b58 b54 b53 b51 b46 b45 b43 b40 b38 b37 b35 b34 b32 b29 b25 b23 b22 b21 b20 b16 b14 b13 b10 b9 b7 b6 b5 b4 b2 b1 b0 r46 r45 r42 r41 r39 r38 r37 r36 r34 r33 r32
c33	b61 b57 b53 b52 b50 b45 b44 b42 b39 b37 b36 b34 b33 b31 b28 b24 b22 b21 b20 b19 b15 b13 b12 b9 b8 b6 b5 b4 b3 b1 b0 r47 r45 r44 r41 r40 r38 r37 r36 r35 r33 r32
c32	b60 b56 b52 b51 b49 b44 b43 b41 b38 b36 b35 b33 b32 b30 b27 b23 b21 b20 b19 b18 b14 b12 b11 b8 b7 b5 b4 b3 b2 b0 r46 r44 r43 r40 r39 r37 r36 r35 r34 r32

A.6 Cable equalization network

An equalization network having the characteristics listed in table A.7 is used for each differential output signals xx_Out_p and xx_Out_n. The equalization network specified in 16.1 is optimized for a 50 meter, 150 Ω twin-ax cable. An example of an resistor-capacitor type equalization network is shown in figure A.5 with a plot of its frequency response.

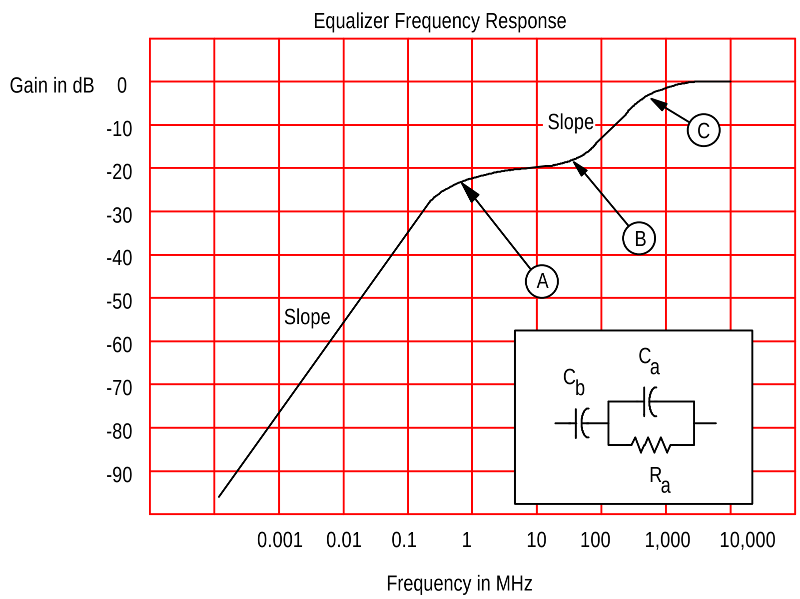


Figure A.7 – Frequency response of RC equalizer network

Table A.7 – Copper cable equalization network

Parameter	Max.	Typical	Min.	Units	Comments
Ca		4		pF	± 5%
Cb		100		pF	± 5%
Ra		675		Ω	± 5%
Freq(A)		-3		dB	pole_A @ 2.2 MHz
Freq(B)		-17		dB	zero_B @ 65 MHz
Freq(C)		-23		dB	pole_C @ 620 MHz
Slope		20		dB/decade	@ 100 KHz